SIEMENS

TRANSDATA
DCAM
Program Interfaces

Reference Manual

Edition January 1985

Software Product DCAM V8.0A (BS2000)

Order No. U1786-J-Z75-1-7600 Printed in the Federal Republic of Germany 1330 AG 3860.2 (1660)	Order No.: U1786-J-Z75-1-7600	A ring binder for this manual may be obtained at a cost of DM 4,- stating the order number U1312-J-Z18-1

68 293

SIEMENS

TRANSDATA
DCAM
Program Interface

Reference Manual

Edition January 1985

Software Product DCAM V3.0A (BS2000)

Order No.:
U1786-J-Z75-1-7600

A ring binder for this manual may be obtained as a spare part by stating the order number U1212-J-Z78-1

TRANSDATA

DCAM
Program Interfaces

Reference Manual

Edition January 1985

Software Product DCAM V8.0A (BS2000)

Order No. U1786-J-Z75-1-7600
Printed in the Federal Republic of Germany
1330 AG 9850.3 (1660)

This document shall not be duplicated, nor its contents used for any purpose, unless express permission has been granted.

Performance features may be added, modified or omitted in the course of product development. Other information in this publication is subject to the same conditions.

Siemens Aktiengesellschaft

PREFACE

This description of the DCAM (Data Communication Access Method) program interfaces is intended for various user groups:

- O and M specialists and application engineers who want a guide to the scope and capabilities of the interface,
- programmers (ASSEMBLER, COBOL) wanting to acquire a basic knowledge of the subject in order to understand and use the more detailed information contained in the programming manuals,
- system and network administrators who do not need to be experts on the interface but would like to have a general knowledge of DCAM.

All readers should be familiar with both the theory and practical application of BS2000. The use of DCAM also presupposes a knowledge of either ASSEMBLER or COBOL.

The manual is divided into four parts:

1. Position of DCAM in BS2000, range of functions and supported hardware.
2. Explanation of basic concepts and points for consideration when planning programs or program systems.
3. Description of the functions of all the DCAM calls and notifications relating to the existence of the DCAM application: virtual connection, data transmission.
4. Outline of the coding of DCAM programs in ASSEMBLER and COBOL, and of the execution of such programs.

Chapters 1 and 2 comprise a general introduction, an overview and information on program planning and organization. Because of the detailed functional descriptions, chapter 3 will be of interest to all users requiring detailed information. Chapter 4 is intended in particular for application engineers; programmers will find it useful as an introduction to the relevant Programmer's Guide. The layout of chapter 3 is identical to that of the same chapters in the User's Guides for ASSEMBLER and COBOL; this is intended to aid the programmer who wishes to refer to the functional descriptions.

Since a large number of new concepts have been introduced, a glossary of TRANSDATA terms has been included at the end of the manual.

The present manual is part of the overall description of the Data Communication Access Method. Related literature includes informative brochures on computer networks and teleprocessing with BS2000, and an outline description of TRANSDATA DCM, as well as the "DCAM Macros" and "DCAM COBOL Calls" User's Guides. Separate manuals are available on generation and administration, and the programming of communication computers and terminals.

The changes made since the previous manual are given in List of Amendments 1.

Other publications are referred to in the text by their "short titles"; the full titles are listed in the bibliography.

To help us continue to improve our manuals, please send us your comments, requests and suggestions using the pink reply forms provided.

Editorial Department D ST PM 2

U1786-J-Z75-1-7600

LIST OF AMENDMENTS 1

The following amendments have been made since publication of the
Revision October 1982 (DCAM V7.0)
and incorporated in the
Edition January 1985 (DCAM V8.0):

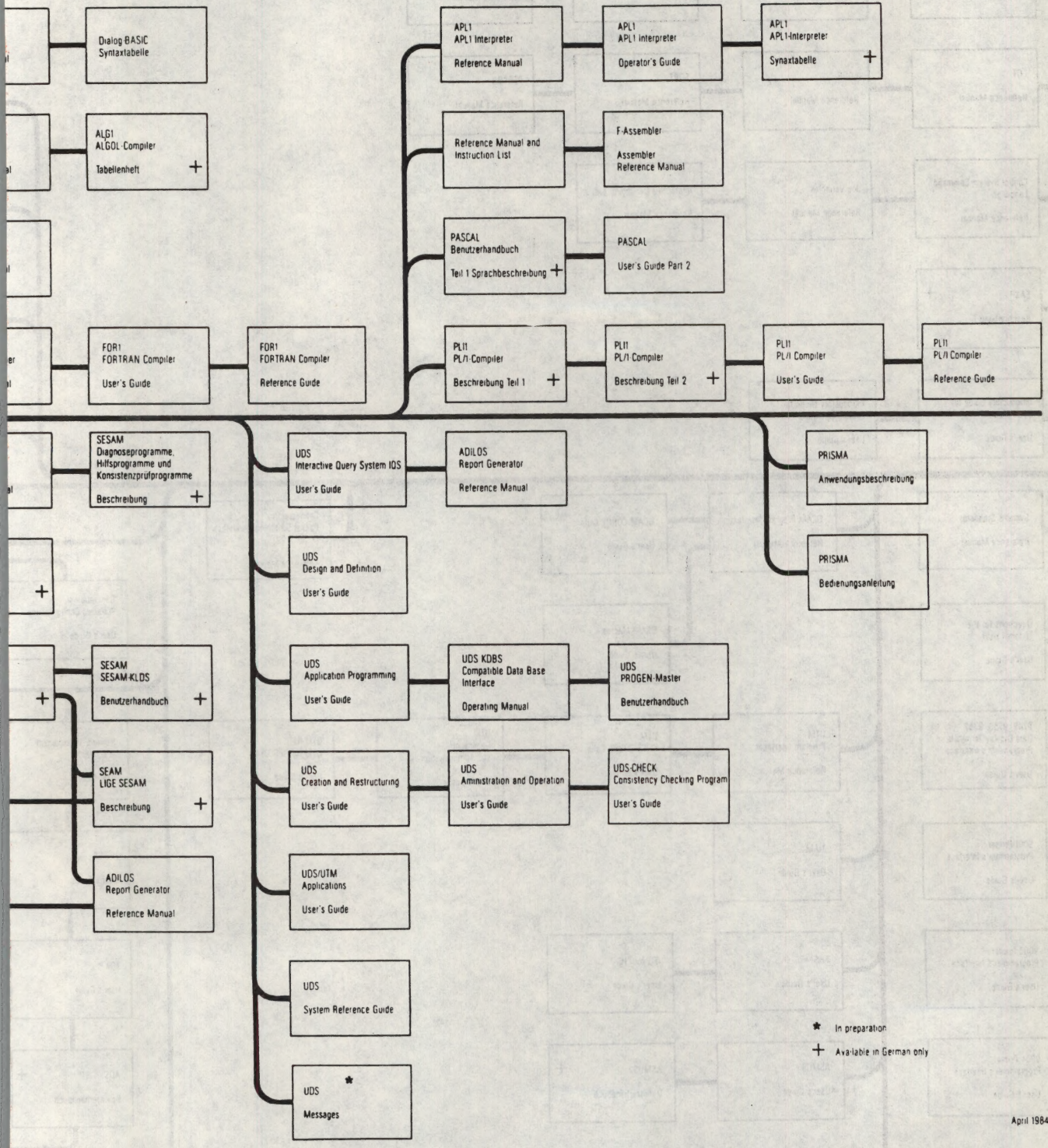
Page	Amendment
2-15, 3-13, 3-20 (Fig. 3-10)	The application can also send a connection message to the partner when a connection is accepted.
2-12, 3-16, 3-29	A GO signal may be issued (SIGNAL) following the overloading of a connection.
3-16	Specification of the maximum length of the data to be transmitted (MAXLN) for optimum system buffer utilization.
3-3, 3-5	To use certain new functions in this version the user must specify DCAMVER when opening a DCAM application.
3-17 (Fig. 3-8)	New message editing options: EXTEND, NLOGC, LACK
2-10, 2-12, 3-3, 3-5, 3-6, 3-32, 3-36	The restriction on the processing of asynchronous notifications under COBOL no longer applies.

BS2000

- System Messages — Reference Manual
- Systems Standard — Reference Manual
- DMS Tape Processing — Reference Manual
- DMS Disk Processing — Reference Manual
- Executive Macros — Reference Manual
- LEASY — Reference Manual
- RFA — Reference Manual
- LMS — Reference Manual
- Laser Printer — Reference Manual
- ARCHIVE — Reference Manual
- Utility Routines — Reference Manual
- EDT — Reference Manual
- EDOR — Reference Manual
- SORT — Reference Manual
- PERCON — Reference Manual
- Control System Command Language — Reference Manual
- Job Variables — Reference Manual
- Interactive Debugging Aid — Reference Manual
- EASP + — Beschreibung
- Introductory Guide for System Users — User's Guide
- Introductory Guide for System Operators — User's Guide

TRANSDATA

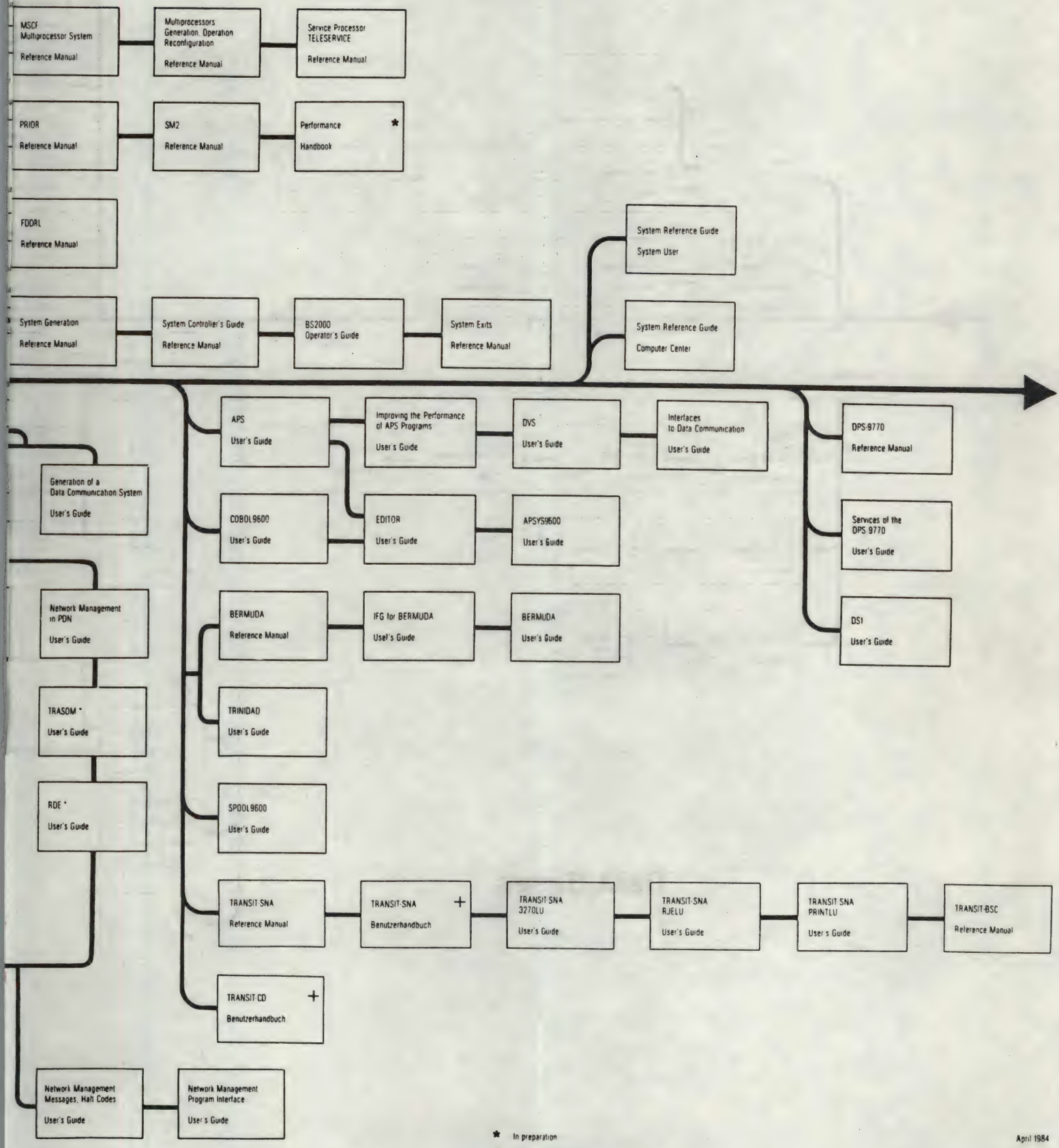
- Systems Standards — Reference Manual
- Directions for the Terminal User — User's Guide
- 9749, 9750, 9752 Data Display Terminals Programmer's Interface — User's Guide
- 9001 Printer Programmer's Interface — User's Guide
- 9003 Printer Programmer's Interface — User's Guide
- 9004 Printer Programmer's Interface — User's Guide
- DCAM Program Interfaces — Reference Manual
  - DCAM COBOL Calls — User's Guide
  - DCAM Macros — User's Guide
- UTM Program Interfaces — Reference Manual
  - UTM KDCS COBOL Calls — User's Guide
    - UTM Generation and Administration — User's Guide
      - UTM-RPC1 UTM for RPG Users — User's Guide
- TIAM — User's Guide
- FHS — User's Guide
  - IFG for FHS — User's Guide
- ASMUS — User's Guide
  - ASMUS2 + — Benutzerhandbuch
- RCOM — User's Guide
  - RCOM-S — Reference Manual
    - RCOM FTP — Reference Manual
- File Transfer — User's Guide
- Connection to Packet Switching Networks (X.25, X.29) — Reference Manual
- Generation of Preferred Configurations — User's Guide
- Network Management in BS2000 — User's Guide
- NEOSP — User's Guide
- TDA * — User's Guide
- ADS + — Benutzerhandbuch
- RBAS — User's Guide
- OMNIS — User's Guide


Dialog-BASIC
Syntaxtabelle
ALG1
ALGOL-Compiler
Tabellenheft +
FOR1
FORTRAN Compiler
User's Guide
FOR1
FORTRAN Compiler
Reference Guide
APL1
APL1 Interpreter
Reference Manual
APL1
APL1 Interpreter
Operator's Guide
APL1
APL1-Interpreter
Synaxtabelle +
Reference Manual and
Instruction List
F-Assembler
Assembler
Reference Manual
PASCAL
Benutzerhandbuch
Teil 1 Sprachbeschreibung +
PASCAL
User's Guide Part 2
PL1I
PL/I-Compiler
Beschreibung Teil 1 +
PL1I
PL/I Compiler
Beschreibung Teil 2 +
PL1I
PL/I Compiler
User's Guide
PL1I
PL/I Compiler
Reference Guide
SESAM
Diagnoseprogramme,
Hilfsprogramme und
Konsistenzprüfprogramme
Beschreibung +
UDS
Interactive Query System IQS
User's Guide
ADILOS
Report Generator
Reference Manual
PRISMA
Anwendungsbeschreibung
PRISMA
Bedienungsanleitung
UDS
Design and Definition
User's Guide
SESAM
SESAM-KLDS
Benutzerhandbuch +
UDS
Application Programming
User's Guide
UDS KDBS
Compatible Data Base
Interface
Operating Manual
UDS
PROGEN-Master
Benutzerhandbuch
SESAM
LIGE SESAM
Beschreibung +
UDS
Creation and Restructuring
User's Guide
UDS
Administration and Operation
User's Guide
UDS-CHECK
Consistency Checking Program
User's Guide
ADILOS
Report Generator
Reference Manual
UDS/UTM
Applications
User's Guide
UDS
System Reference Guide
UDS
Messages
* In preparation
+ Available in German only

April 1984

Publications Plan for

- BS2000
- TRANSDATA
- Programming Systems
- Data Bases

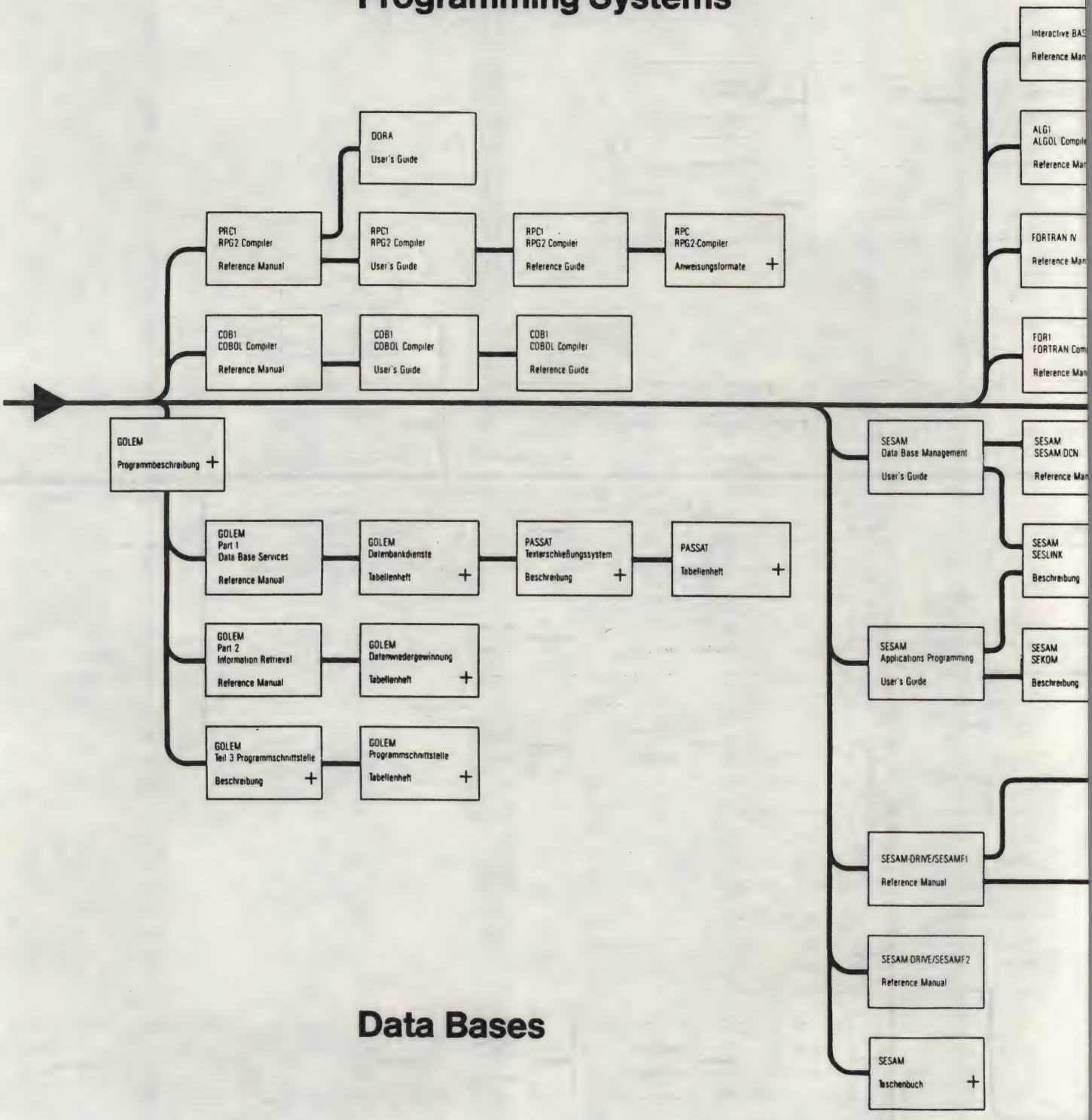


* In preparation

\+ Available in German only

April 1984

Programming Systems



Data Bases

CONTENTS

1	**DCAM in the TRANSDATA Data Communication System (General Introduction)**	**1-1**
1.1	TRANSDATA DCM (Data Communication Methods)	1-1
1.2	Traffic Relations in the Data Communication System with DCAM	1-3
1.3	Hardware Spectrum for the Application of DCAM	1-5
2	**Basic Concepts and an Introduction to the Use of the DCAM Interface**	**2-1**
2.1	Basic Concepts	2-1
2.1.1	Program, Data, Task	2-1
2.1.2	Communication Partners	2-2
2.1.3	Symbolic Addressing	2-4
2.1.4	Virtual Connection	2-6
2.2	Characteristic Features of DCAM	2-7
2.2.1	Distribution of Incoming Messages	2-7
2.2.1.1	Originator-Oriented Queue and Common-Receiver Queue	2-7
2.2.1.2	Distribution Code Oriented Queues	2-8
2.2.1.3	Implicit Distribution Code	2-10
2.2.2	Calls and Notifications	2-10
2.2.3	Security Concept	2-10
2.2.3.1	Protection against Unauthorized Access	2-10
2.2.3.2	Data Security	2-11
2.2.3.3	Express Messages	2-12
2.2.3.4	Data Flow Control	2-12
2.3	Planning Program Structure	2-13
2.3.1	Functions of a DCAM Program	2-13
2.3.2	Basic Structure of a DCAM Program	2-15
2.3.3	Control of Primary and Secondary Tasks	2-17
2.3.4	Access to Terminals	2-18
2.3.4.1	Line Terminal	2-19
2.3.4.2	Form Terminal	2-22
2.3.5	Implementation of Distributed Processing	2-23
2.3.5.1	A DCAM Application as a Partner	2-23
2.3.5.2	A UTM Application as a Partner	2-24
2.3.5.3	A PDN Application as a Partner	2-24
3	**DCAM Functions**	**3-1**
3.1	Existence Function	3-1
3.1.1	Opening a DCAM Application	3-2
3.1.1.1	Non-Shareable DCAM Application	3-3
3.1.1.2	Primary Opening of a Shareable DCAM Application	3-4
3.1.1.3	Primary Opening - Use of Distribution Codes	3-5
3.1.1.4	Secondary Opening	3-6
3.1.1.5	Secondary Opening - Use of Distribution Codes	3-6
3.1.2	Altering the State of a DCAM Application	3-7
3.1.3	Testing the State of a DCAM Application	3-8
3.1.4	Closing a DCAM Application	3-8

3.2 Connection Function 3-9
3.2.1 Establishing a Virtual Connection 3-10
3.2.1.1 Definition of the Connection to be Established 3-14
3.2.1.2 Connection Request 3-18
3.2.1.3 Acceptance of a Request 3-20
3.2.1.4 Connection Setup - Use of Distribution Codes 3-21
3.2.1.5 Linking Up to a Predefined Connection 3-22
3.2.2 Interrogation of the Entries on Partners and Connections . 3-24
3.2.3 Rejection of a Connection Request 3-25
3.2.4 Changing the Characteristics of a Connection 3-25
3.2.5 Deletion of a Request 3-26
3.2.6 Closing a Connection 3-26
3.3 Data Transmission Function 3-27
3.3.1 Transmission of a Message 3-28
3.3.2 Receiving a Message or Transport Acknowledgment 3-32
3.3.3 Combined Transmitting and Receiving 3-36
3.3.4 Cancelling of Receive Macros and Changing the Connection State CS/CA 3-36
3.3.5 Control of Distribution Code Assignment 3-37
3.3.5.1 Assignment of a Distribution Code Name to Distribution Code Groups 3-38
3.3.5.2 Cancellation of the Assignment 3-39
3.4 Name Assignment Function 3-40

4 Coding and Execution of DCAM Programs 4-1
4.1 Writing an ASSEMBLER Program 4-1
4.1.1 Macro Calls and Control Blocks 4-1
4.1.2 Synchronous Execution of DCAM Calls 4-6
4.1.3 Asynchronous Execution of DCAM Calls 4-7
4.1.4 Feedback Information 4-10
4.1.5 Asynchronous DCAM Notifications 4-11
4.1.6 Reentrant ASSEMBLER Programs 4-14
4.1.7 Use of Other System Interfaces 4-16
4.2 Generation of a COBOL Program 4-17
4.2.1 COBOL Calls and Data Structures 4-17
4.2.2 Execution of the Calls 4-20
4.2.2.1 Synchronous Execution 4-20
4.2.2.2 Asynchronous Execution 4-20
4.2.3 Feedback Information 4-21
4.2.4 Reentrant COBOL Programs 4-22
4.2.5 Use of Other System Interfaces 4-25
4.3 Execution of a DCAM Program: DCAM Task 4-26
4.3.1 Starting a DCAM Task 4-26
4.3.2 Termination of a DCAM Task 4-27

A Appendix A1-1
A.1 DCAM Calls A1-1
A.2 Limit Values A2-1
A.3 Setting Up a Connection from a Terminal A3-1

Glossary of TRANSDATA Terms
Literature

List of Figures and Tables

Figure		Page
1-1	Structure of TRANSDATA DCM	1-2
1-2	DCAM traffic relations	1-4
1-3	TRANSDATA hardware spectrum with BS2000	1-6
1-4	Virtual terminal support	1-8
2-1	The task as an administrative unit in BS2000	2-2
2-2	Examples of DCAM applications	2-3
2-3	Example of network configuration with symbolic addresses	2-5
2-4	Example of virtual connections	2-6
2-5	Example of queue access	2-8
2-6	Example of access to distribution code-oriented queues	2-9
2-7	Example of message acknowledgment	2-11
2-8	Schematic layout of a DCAM program	2-16
2-9	Transmitting and receiving data to be represented in line format	2-19
2-10	Transmitting and receiving formatted data	2-22
3-1	DCAM application states	3-7
3-2	Connection request	3-10
3-3	LOGON notification; request testing and acceptance	3-11
3-4	Partner information inquiry and request acceptance	3-11
3-5	Acceptance of a request	3-12
3-6	PROCON notification request to proposed partner and acceptance	3-12
3-7	Agreement on characteristics	3-14
3-8	Message editing options	3-17
3-9	Processing a request	3-19
3-10	Acceptance of a request	3-20
3-11	Description of the formats and values of distribution codes	3-22
3-12	Linking up to a predefined connection	3-23
3-13	Message to a line terminal	3-30
3-14	Message layout for EDITOUT=PHYS	3-31
3-15	Example of message structuring	3-31
3-16	Specifications for receiving a message	3-33
3-17	Summary of message distribution with distribution codes (example)	3-37
3-18	Example of task group addressing with distribution code	3-38
3-19	Example of assignment of a DCAM application name	3-40
4-1	ACB-related macro calls	4-2
4-2	RPB-related macro calls	4-3
4-3	Example of control block addressing without identifiers	4-5
4-4	Example of the addressing of an RPB in which valid identifiers were entered	4-5
4-5	Example of synchronous execution of the YRECEIVE call	4-6
4-6	Example of asynchronous execution of the YRECEIVE call	4-7
4-7	Example with contingency routine	4-8
4-8	Example of express messages being received via an asynchronous DCAM notification (EXPR)	4-13
4-9	Models for shared code programming	4-15
4-10	Segmentation of a COBOL program	4-23
4-11	Generation of shared code	4-24
4-12	Task types	4-26
4-13	Timesharing and inquiry and transaction processing in a task	4-27

Table		Page
2-1	Transaction processing with BS2000	2-14
2-2	Control of a primary task	2-17
3-1	Definition of non-shareable DCAM applications	3-3
3-2	Description of shareable DCAM applications opened by the primary task	3-4
3-3	Description of shareable DCAM applications opened by a secondary task	3-6
3-4	Handling of excess-length messages	3-15
3-5	Message code	3-15
3-6	Inquiry variants	3-24
3-7	Definitions for the transmission of a message	3-29
3-8	Message distribution and receiving	3-32
3-9	Distribution of transport acknowledgments	3-35
4-1	Feedback information format	4-10
4-2	Asynchronous DCAM notifications	4-11
4-3	Shared code program structure	4-15
4-4	Subroutines and data structures	4-18
4-5	Status testing	4-19

General Introduction to DCAM 1

Basic Concepts and an Introduction to Application 2

DCAM Functions 3

Coding and Execution of DCAM Programs 4

Appendix A

U1786-J-Z75-1-7600

1 DCAM IN THE TRANSDATA DATA COMMUNICATION SYSTEM ‹GENERAL INTRODUCTION›

Within the TRANSDATA data communication system, access methods such as DCAM are located at the interface to the "outside", i.e. to the user. DCAM provides the user with all the capabilities of TRANSDATA at his program interface. At the same time, DCAM is integrated in the TRANSDATA DCM (Data Communication Methods) communication access system along with all other BS2000 access methods.

1.1 TRANSDATA DCM ‹DATA COMMUNICATION METHODS›

In BS2000 Version 4, access to the data communication system has been implemented by a standard system component. As is the case with the TRANSDATA components, this component has a multi-level structure. This results in clear-cut interfaces, thus ensuring that subsequent modifications in the hardware configuration or the system software will not affect the user interface. The user will therefore not be faced with extra costs of reorganization when future developments offering further advantages become available. At present, TRANSDATA DCM provides the user with the following features:

- **Use of DCAM** for handling transaction processing;
- support of terminal programming through the application of **virtual terminals**;
- administration at optional operator consoles, also, if desired, in a DCAM program using the **multiconsole operation module (UCON)**;
- powerful support of the established interfaces for **timesharing (RTIO)** and **remote batch processing (RBP)**.

This high level of convenience offered by the user interface is made possible by the modules provided in TRANSDATA for the user services:

- **DCAM (Data Communication Access Method)** for the implementation of the DCAM interface in ASSEMBLER and COBOL;
- **VTSU (Virtual Terminal Support)** for the implementation of the virtual terminals;
- **TIAM (Terminal Interactive Access Method)** for the implementation of the RTIO ASSEMBLER interface (remote terminal input output);
- **RBAM (Remote Batch Access Method)** for batch mode support with RBP (remote batch processing) commands.

These modules, in turn, have a defined interface to **BCAM (Basic Communication Access Method)**. The tasks common to all access methods, such as transport management, buffering, operation of the channel trunk to the front-end processor, etc. are implemented in this basic module.

Fig. 1-1 contains an overview of the TRANSDATA DCM components.

The DCAM opens up the wide-ranging potential of the data communication system, especially unrestricted data communication with all communication partners (applications and terminals), to the user.

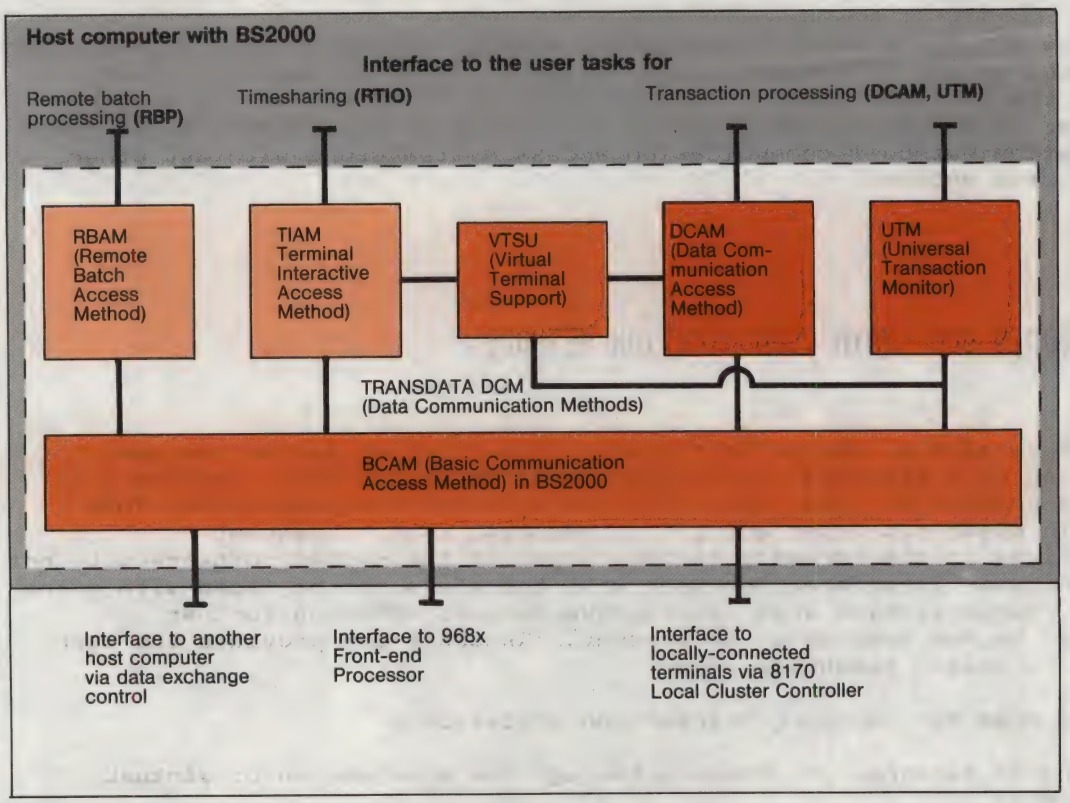


Fig. 1-1 Structure of TRANSDATA DCM

1.2 TRAFFIC RELATIONS IN THE DATA COMMUNICATION SYSTEM WITH DCAM.

DCAM enables one or more tasks (programs) in the host computer to communicate with a single terminal or with a number of terminals. This would represent the classic traffic pattern known from the conventional star-type network where the processing capacity is concentrated in a single computer. However, DCAM also supports the principle of distributed processing power by means of computer-to-computer communication. The tasks (programs) of one computer can communicate with the tasks (programs) of another computer. Moreover, tasks (programs) in the same host computer can also communicate with each other via DCAM. This is especially important for the test phase of such programs, as terminals can be easily simulated by programs. The traffic relations are shown in Fig. 1-2.

Access to the "$CONSOLE" system application is also possible within the user computer, thus enabling program-based operation and administration of the system. If the appropriate commands are issued to the program from a terminal, the program can pass them on to "$CONSOLE" (application for multiconsole operation) for execution. Thus operation and administration via a terminal is possible.

A paradigm for such a program is shown in chapter 4 of "DCAM COBOL Calls". Further information on this subject is to be found in the manuals "Operator's Guide" and "Generation of a Data Communication System".

New traffic relations have been made possible by an extended hardware spectrum which is supported by BS2000 Version 4 and up.

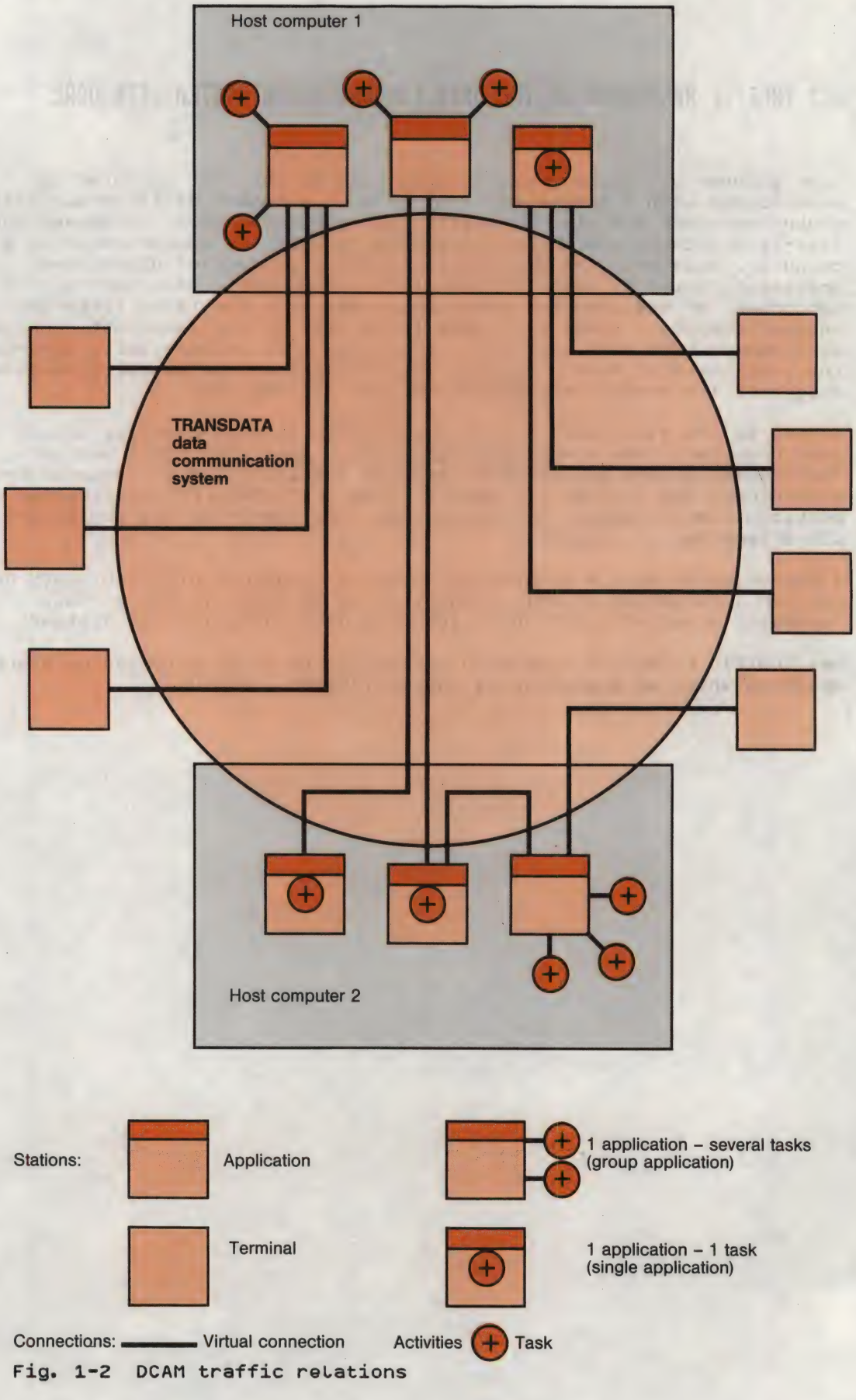


Fig. 1-2 DCAM traffic relations

1.3 HARDWARE SPECTRUM FOR THE APPLICATION OF DCAM

BS2000 Version 4 permits the use of a large number of powerful hardware modules in the TRANSDATA data communication system. Fig. 1-3 contains an overview of all applicable devices. A new feature is the support of the 966x Terminal Computer of the TRANSDATA computer family.

The 8170 Local Cluster Controller is used for the first time. This enables terminals to be connected directly to the host computer.

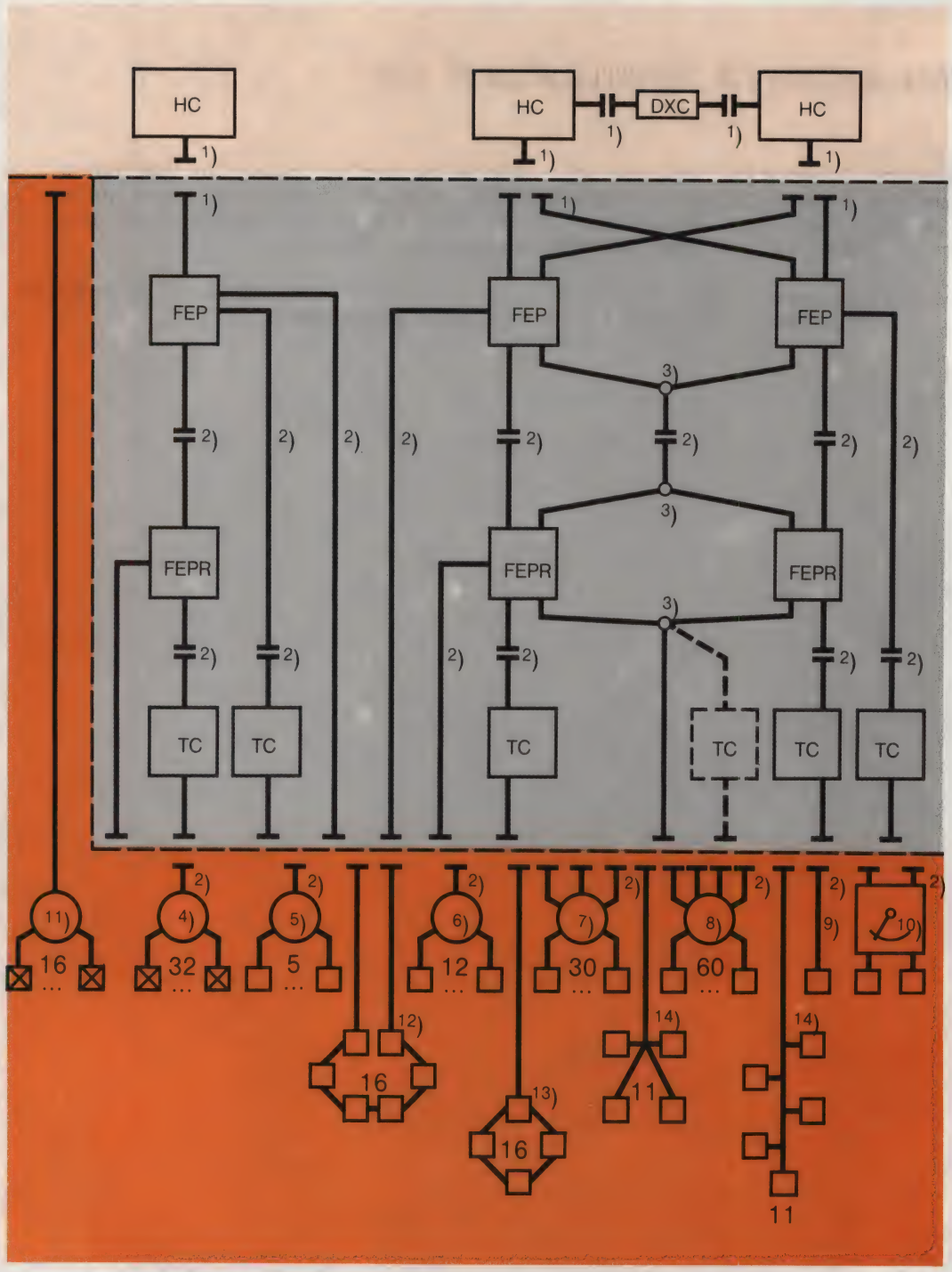


Fig. 1-3 TRANSDATA hardware spectrum with BS2000

Legend:

HC = Host Computer
DXC = 4627 Data Exchange Control
FEP = 968x Front-end Processor
FEPR = 967x Remote Front-end Processor
TC = 966x Terminal Computer

1) Channel trunk

2) Direct connection or data transmission line

3) Dual system switchover unit - alternate connection

4) 8171 Remote Cluster Controller

5) 8906 Interface Expander

6) 8901 Medium-speed Concentrator

7) 8902 Medium-speed Concentrator

8) 8903 Medium-speed Concentrator

9) Dedicated circuit

10) Dial circuit (TELEX, DATEX, telephone network)

11) 8170 Local Cluster Controller

12) Local loop

13) Relocated local loop

14) Multipoint network using data transmission facility (Modem N10)

Terminals (including TRANSDATA 810 Terminal System)

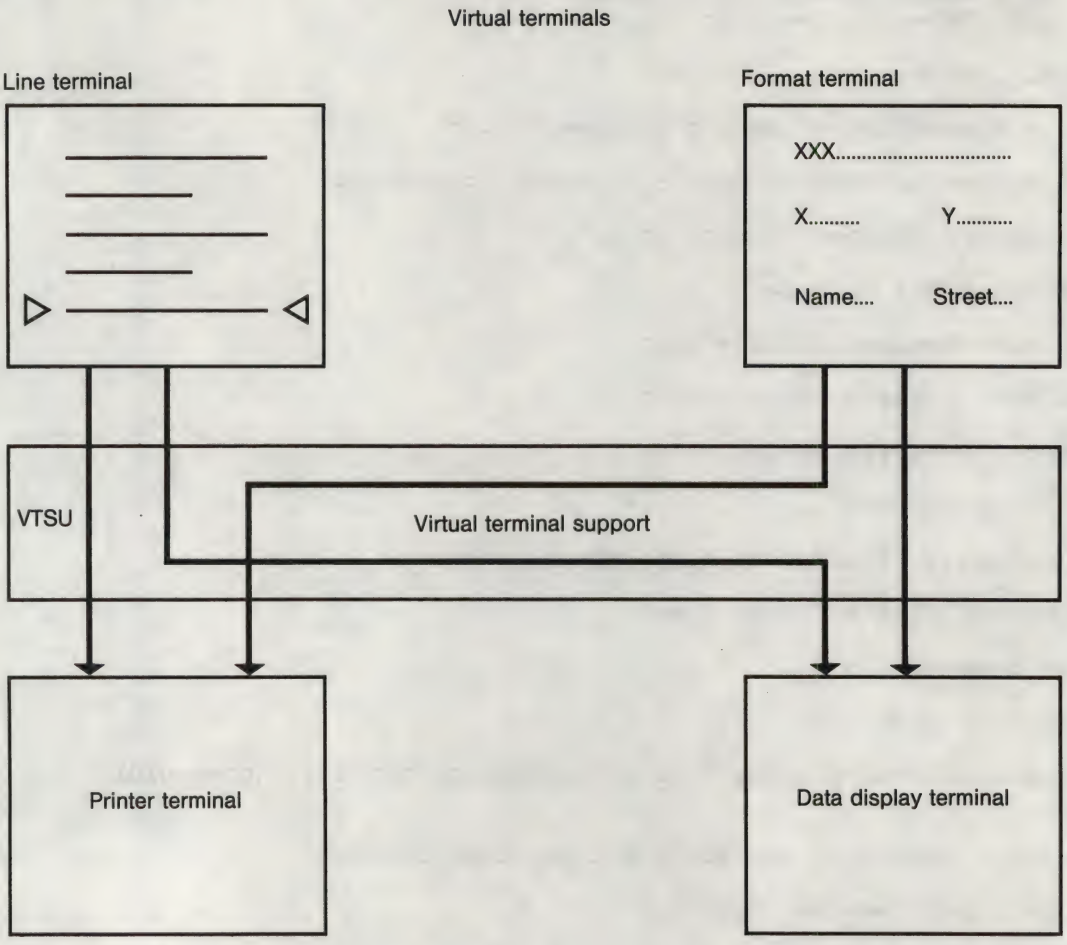
TRANSDATA 810 Terminal System only

The number denotes the maximum number of connections.

7.500 and
7.700 systems

TRANSDATA 960 System

As previously mentioned, the user is supported by the **virtual terminal** feature when programming the terminals (see 2.3.4). Fig. 1-4 shows which which devices this applies to.



Model numbers of actual terminals

8103
8110[1)]
8110[2)]
8122[2)]

1) not supported as format terminal

2) connectable via 8112

8150[3)] 8160[4)]
8151[3)] 8161[4)]
8152[3)] 8162[4)]
8153[3)] 975x[5)]

Additional hardcopy devices

3) 8100,8120

4) 8121, 8122,9002,9003

5) 9001,9004

Fig. 1-4 Virtual terminal support

2 BASIC CONCEPTS AND AN INTRODUCTION TO THE USE OF THE DCAM INTERFACE

DCAM programs differ from other programs not only in that they use the DCAM interface, but access to the TRANSDATA data communication system also determines their structure. Therefore, it is the object of this chapter to supply answers to the following questions:

- Which characteristics determine the **performance capability** of the DCAM interface?
- What is the **basic structure** of a DCAM program?
- What points have to be considered when **planning** a DCAM program?

Prior to this, however, some basic concepts will be discussed to facilitate understanding.

These are:

- program, data, task
- communication partner
- symbolic addressing
- virtual connection

2.1 BASIC CONCEPTS

2.1.1 Program, Data, Task

In BS2000, the program and data areas and the task controlled by the program are generally administered as a single unit. For the sake of clarity, no distinction will be made between these three components in the following text. It is always this unit that is being referred to whenever a program or task is mentioned. The task's data area is important in the transmitting and receiving of messages since it is to and from this area that messages are transmitted. That is why the data area is referred to explicitly in certain passages.

The specific structural and organizational characteristics of a DCAM program are dealt with in a separate section (2.3).
Another section describes the particular situation that arises when a program controls several tasks (shareable programs/modules) (see 4.1.6).
Program execution - the DCAM task - is also described in a separate section (see 4.1.6 and Fig. 2-1).

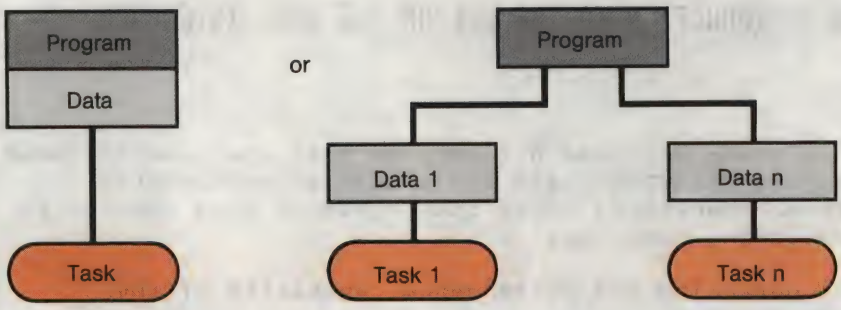


Fig. 2-1 The task as an administrative unit in BS2000

2.1.2 Communication Partners

In the TRANSDATA data communication system there are two different types of communication partner:

terminals and applications.
Applications can be:

- DCAM applications,
- UTM applications,
- PDN applications.

The programmer has no influence over the existence of terminal-type communication partners. Such communication partners are generated or removed either at communication system generation time or by means of administration instructions issued by the network administrator. The generation of DCAM application type communication partners is a function of the DCAM interface.

Those users of the TRANSDATA data communication system who can set up virtual connections to other partners are referred to as communication partners in a wider sense. In the more restricted sense, this name is used to denote the two partners linked by an existing virtual connection.

The **terminal** is characterized by its technical implementation on the one hand and by the person operating it on the other. The interaction of these two elements, i.e. the device and the user of the device, implements the terminal communication partner at a given point in time. This communication partner can only maintain virtual connections to other partners and transmit and receive data via these connections with the means provided by the data communication system.

The **DCAM application** exists exclusively in a host computer controlled by BS2000. It is defined by at least one DCAM program and is generated by the data communication access method. It is administered by the communication access system as an addressable unit for incoming or outgoing messages and notifications and is canceled by program request. The DCAM application is the address at which tasks (programs) or groups of tasks are known to the data communication system. They thus become communication partners.

A DCAM application can be **opened by one task** (non-shareable) or **by several tasks**, i.e. a **task group** (shareable). The first opening task is the primary task. It may be followed by secondary tasks if required. Several DCAM applications can be opened in one program. As each of them can be regarded separately, this feature is not discussed at greater length in this manual. The DCAM application's name is established when the application is generated (cf. example in Fig. 2-2).

This **name** can be generated dynamically by the system, or a predefined name known to the data communication system can be used. The name of the DCAM application must be unique in the host computer in which the DCAM application is generated.

Communication with other distributed processing applications is described in section 2.3.5.

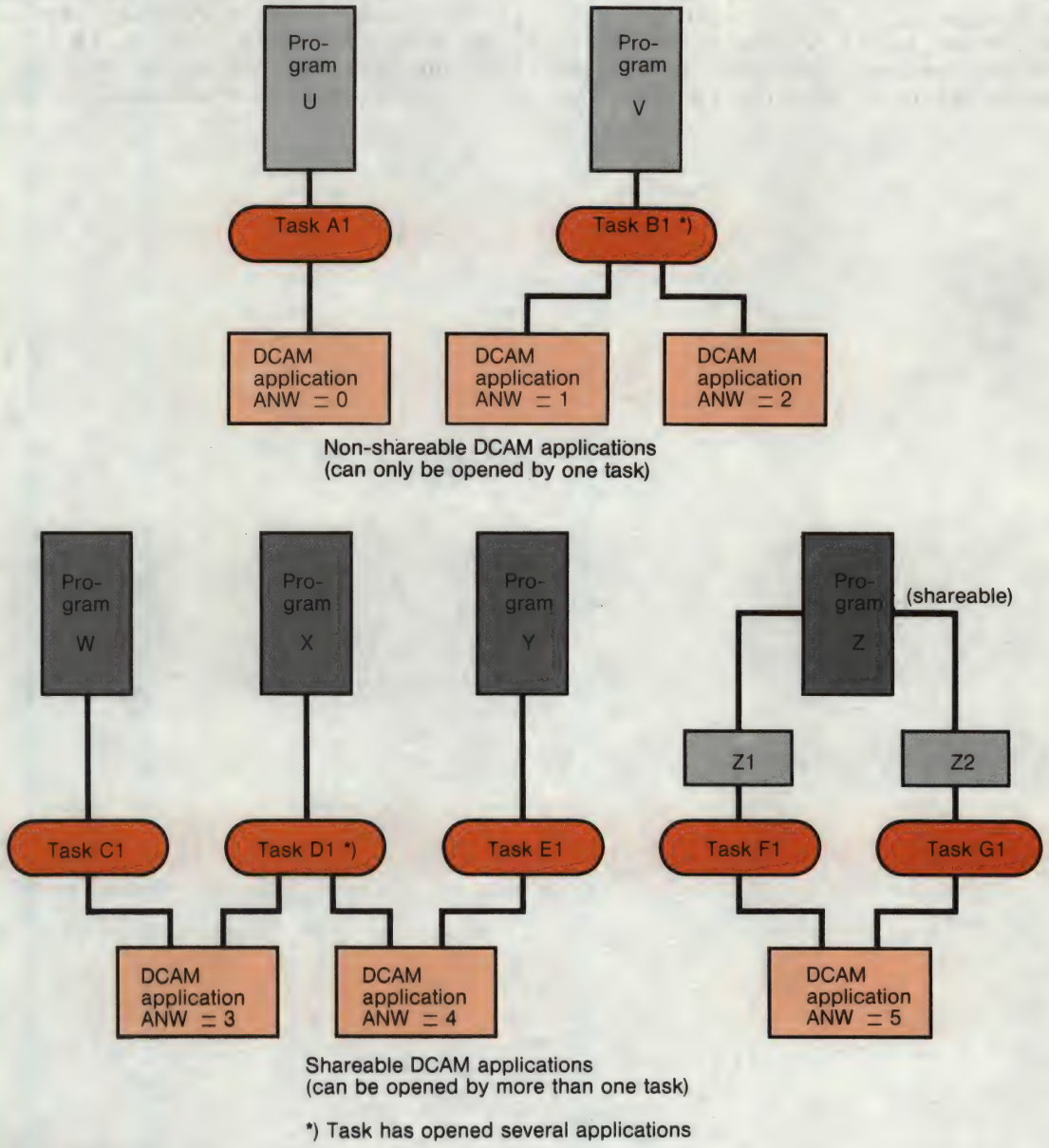


Fig. 2-2 Examples of DCAM applications

2.1.3 Symbolic Addressing

Two names suffice to address a partner: the name of the processor node the partner is connected to as a station, and the name of the communication partner (refer to the Glossary of TRANSDATA Terms).

The names of the terminals and processor nodes are defined when the data communication system is generated. The names of the DCAM applications can be defined in the program when the applications are opened. In specific cases they can also be generated dynamically in the system (cf. Fig. 2-3). The symbolic addresses are converted in stages within the data communication system. Thus, the logical network address is created one level down and the physical address of the device another level below that.

However, the DCAM user only needs to know the symbolic addresses. All names used can have a maximum length of 8 characters. The first character must be alphabetic or $, @, #. Characters 2 to 8 may also be digits from 0...9 (ASSEMBLER name convention). A user application name may not begin with the dollar sign $, as this is reserved for internal system applications.

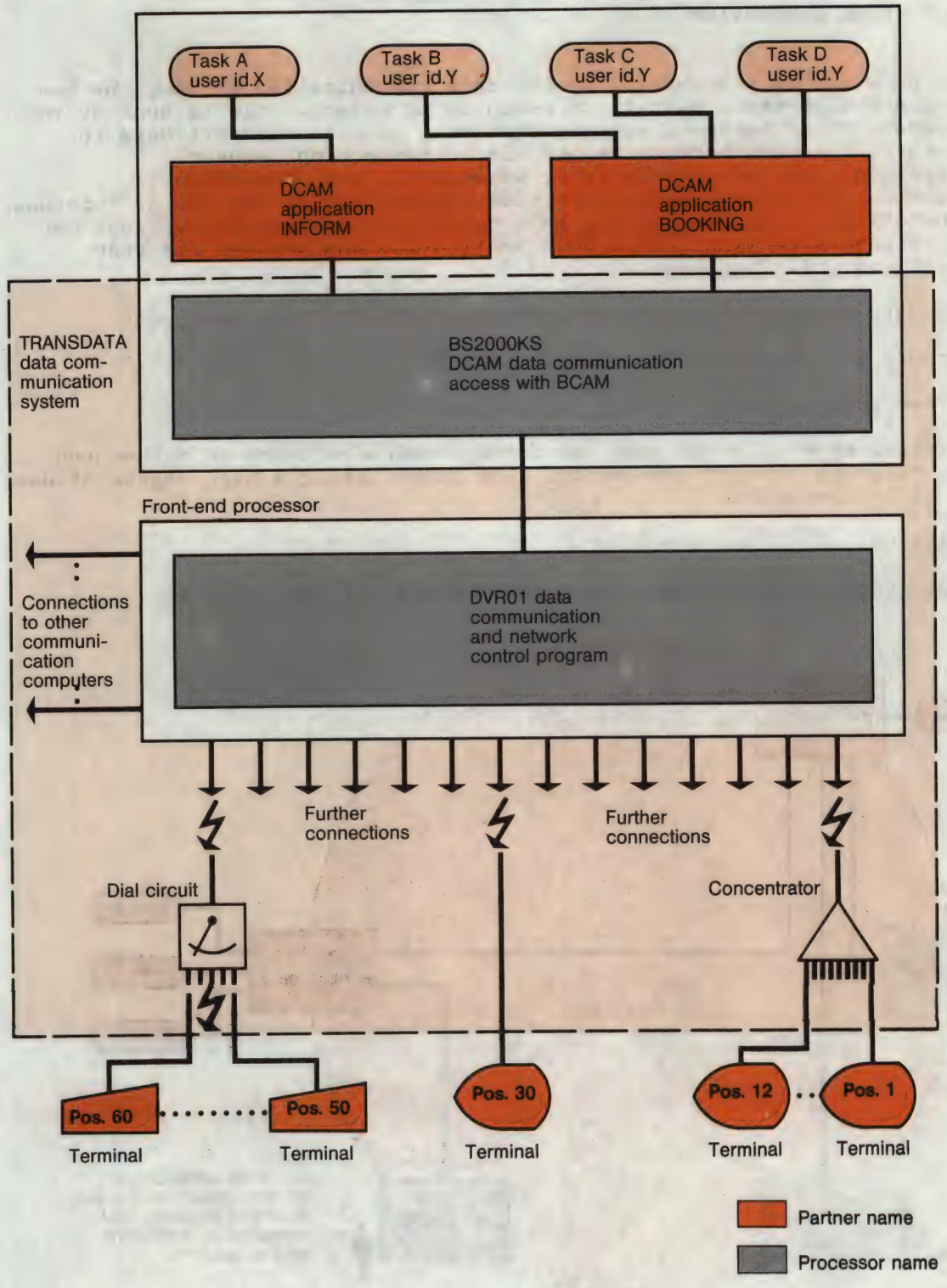


Fig. 2-3 Example of network configuration with symbolic addresses

2.1.4 Virtual Connection

Before data can be transmitted by the data communication system, the two communication partners must first reach an agreement. This is done by the two communication partners stating that they wish to transmit data to each other, i.e. one partner begins with a **connection request (acquisition)**. If the other partner **accepts** it, the connection is established. The data transmission modalities (such as the use of "virtual terminals") are specified during this operation. Virtual connections can exist between a terminal and a DCAM application and between two DCAM applications (cf. Fig. 2-4).

In the data communication system a virtual connection is assigned:

- a route in the network,
- buffer areas, etc.

The virtual connection between two communication partners is documented in the relevant network components, thus guaranteeing a high degree of data transmission security.

Restriction:

Virtual connections between data terminals are not yet possible.

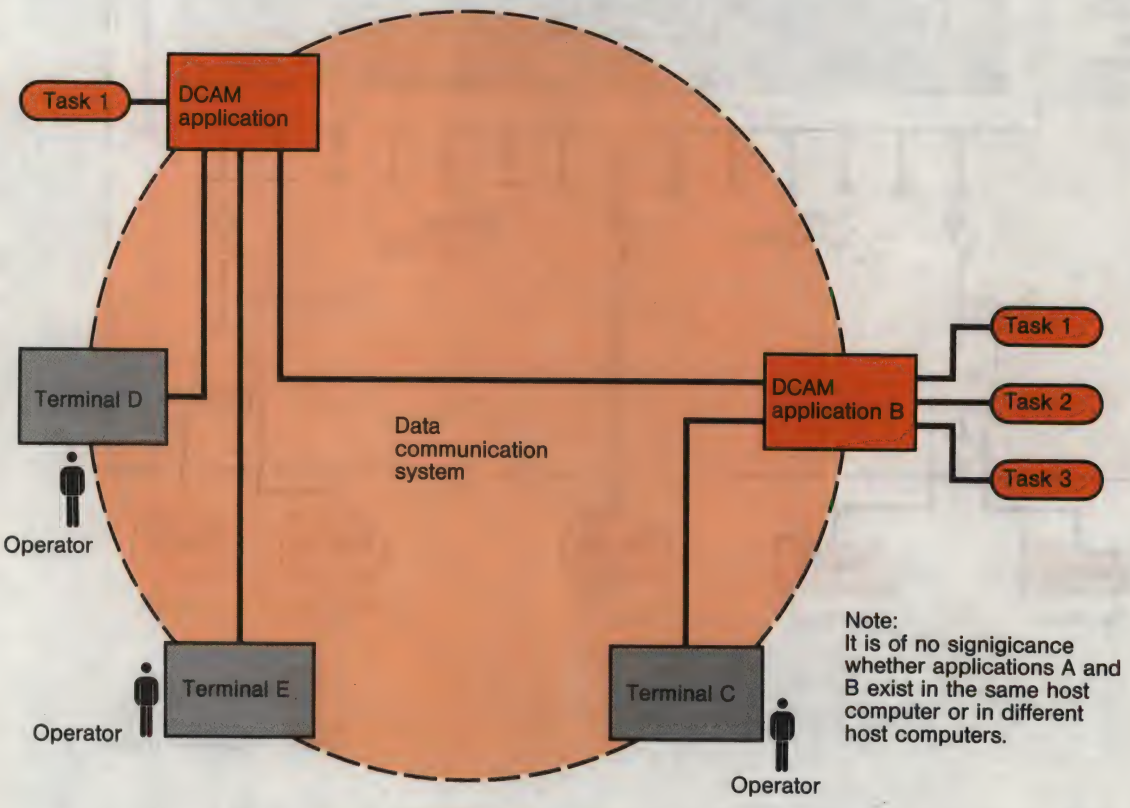


Fig. 2-4 Example of virtual connections

2.2 CHARACTERISTIC FEATURES OF DCAM

2.2.1 Distribution of Incoming Messages

There is a choice of two methods for the distribution of messages arriving for a DCAM application:

- distribution via originator-oriented queues or common-receiver queues
- distribution via distribution code oriented queues

The first method is suitable both for shareable and for non-shareable DCAM applications, the second only for shareable DCAM applications.

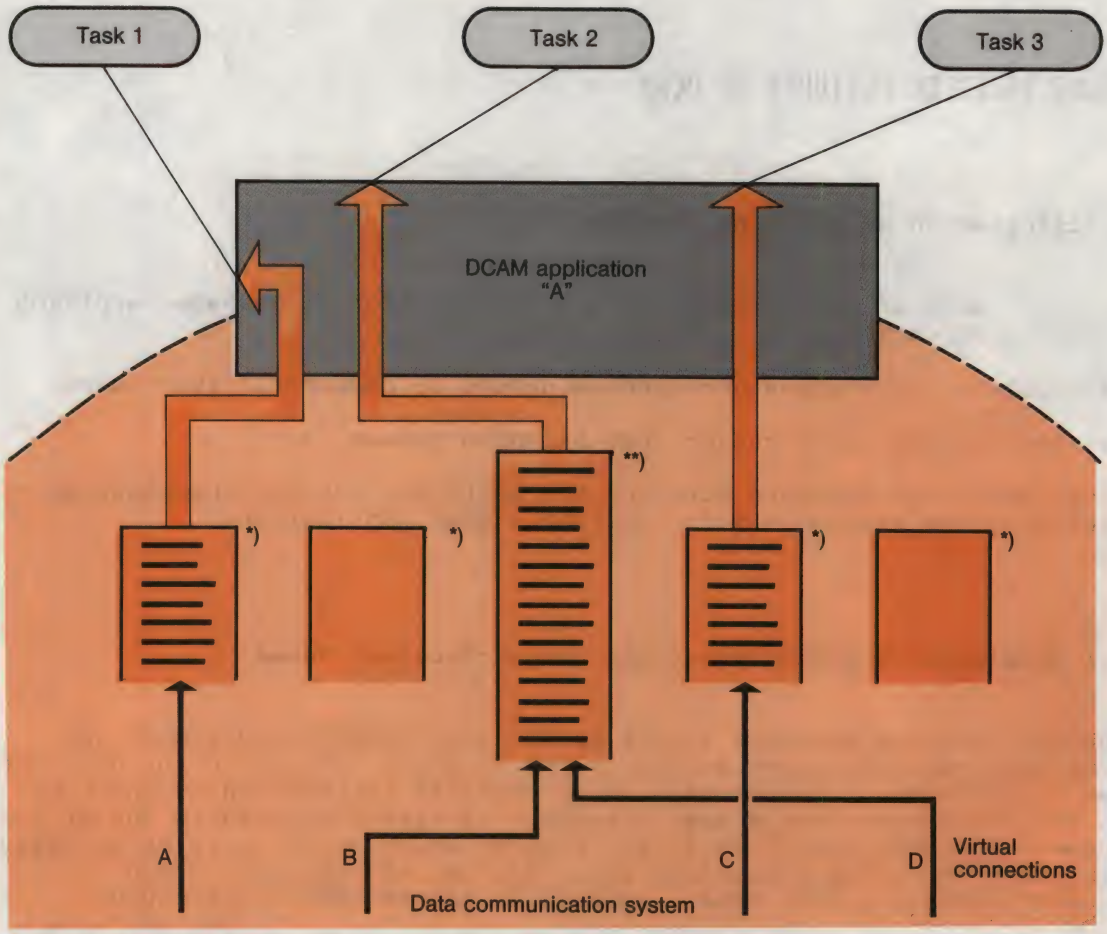
2.2.1.1 Originator-Oriented Queue and Common-Receiver Queue

A queue for incoming messages is set up for each established connection enabling originator-oriented access to the messages. Access in the order of arrival (regardless of the message originator) is implemented by another queue, the common-receiver queue. A message is always entered in one of the two queues. The user specifies in the program which queue is to be accessed. This is possible during connection setup and can be respecified for subsequent processing each time a message is transmitted or received.

In this way, messages can be fetched only from the originator-oriented queue for a definable period of time. In the case of shareable applications, the **Link between a task and a virtual connection** is established in this way. This link is canceled again when the common-receiver queue is used.

Fig. 2-5 provides an overview.

Task 1 has specified (during logon or during the last transmission or reception operation) that it wants to receive messages via connection A, and accesses the associated queue with receive macros. The same applies to task 3 and connection C. Messages from connections B and D are received by task 2 but can also be received by other tasks.



*) Originator-oriented queues
**) Common-receiver queue

Fig. 2-5 Example of queue access

2.2.1.2 Distribution Code Oriented Queues

It is presumed that the task group is controlled by programs with different capabilities, i.e. that each program can only process certain messages. The message/program (task) link is established by means of a **code** which is contained **in the message** itself. The user can select the code and specifies specific codes when the virtual connection is established (for a detailed description of the various options refer to 3.2.1.4). **A separate queue** is created for each **code group**, and the **allocation** of the queues to the programs is **controlled by the primary task** (special macros are provided for this purpose, see 3.3.4). A distribution name is defined for this purpose when the application is opened. It is assigned to the distribution codes in order to link them to a task.

The name can be the same for several tasks, as, for instance, when one program controls more than one task. DCAM then passes on messages to the various tasks in accordance with the FIFO principle (first in - first out: the first entry in the queue will be the first one processed).

When the communication partner enters the code defined at connection set-up time or later, he will reach the task whose distribution name has been assigned to this code by the primary task.

Thus the primary task controls

- the assignment of distribution codes to a connection:
  when setting up a connection, it determines which codes are to be used and it can redefine these codes for an active connection at any time via a new macro call.

- the assignment of the distribution codes to a task by means of the distribution name.

Messages which cannot be delivered because their code is not allocated or is invalid are delivered to the primary task.

Fig. 2-6 shows an example of queues for distribution code groups containing only one code each.
As messages with the same code arrive from different partners, originator-oriented access to these messages is also possible. The distribution name DISTRIB1 is assigned both to task 1 (primary task) and to task 2. At this point the FIFO rule (see above) comes into effect: messages with the codes CA and CF will be passed on in the sequence of their arrival to the next task issuing a receive call. Messages with the codes CD, CX and C$ are distributed to task 1 because they are not assigned to any other task.

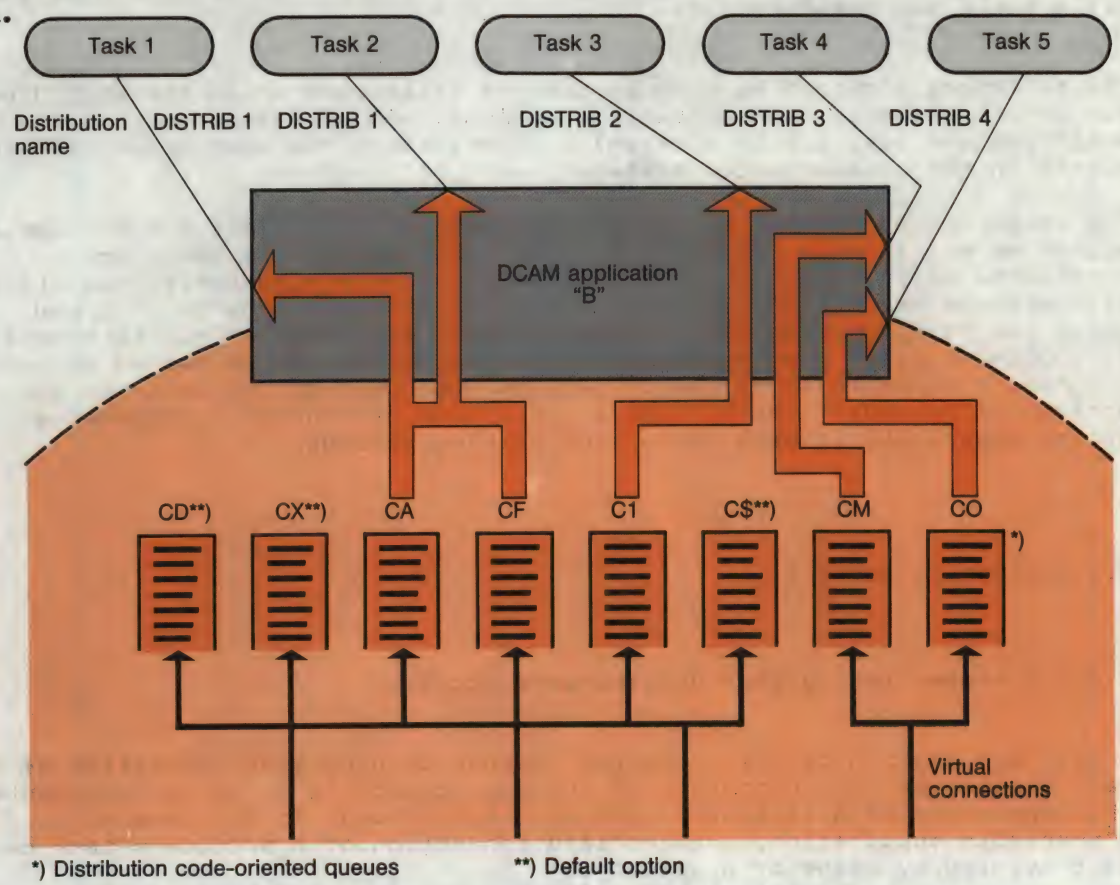


Fig. 2-6 Example of access to distribution code-oriented queues

2.2.1.3 Implicit Distribution Code

If a sequence of messages contain the same distribution code it is sufficient to specify this code in the first message. For this purpose the primary task defines a code indicator character which indicates that a **distribution code** is included in a message or message sequence. The distribution code immediately follows this character. In this case it may be no more than 7 characters long.

If messages without a distribution code are received, they are allocated according to the last distribution code received. For subsequent messages DCAM implicitly assumes the distribution code last in force. This applies until the partner sends a message in which a distribution code is explicitly specified.

If a code indicator character has not been defined by the primary task, then a distribution code must be included in each message. Otherwise they will be delivered to the primary task.

The same applies if the user omits to begin a sequence of messages with at least one message containing a distribution code.

2.2.2 Calls and Notifications

The functions provided by DCAM consist of calls with which the BS2000 user can effect the execution of certain actions, and of asynchronous notifications (see 4.1.5) with which DCAM informs the user about certain events in the communication system.

The calls issued to DCAM all begin with the letter Y. They are available either as **macro calls (ASSEMBLER)** or as **COBOL calls.** The calls are terminated after execution or when a defined processing period has elapsed **(synchronous execution).** It is also possible to have control returned after the call has been issued **(asynchronous execution).** Call termination is indicated by an **asynchronous notification** which can be tested in the program. A separate (contingency) routine can also be initiated by the arrival of the notification (see 4.1.3). This asynchronous processing serves especially to make use of the waiting periods.

2.2.3 Security Concept

2.2.3.1 Protection against Unauthorized Access

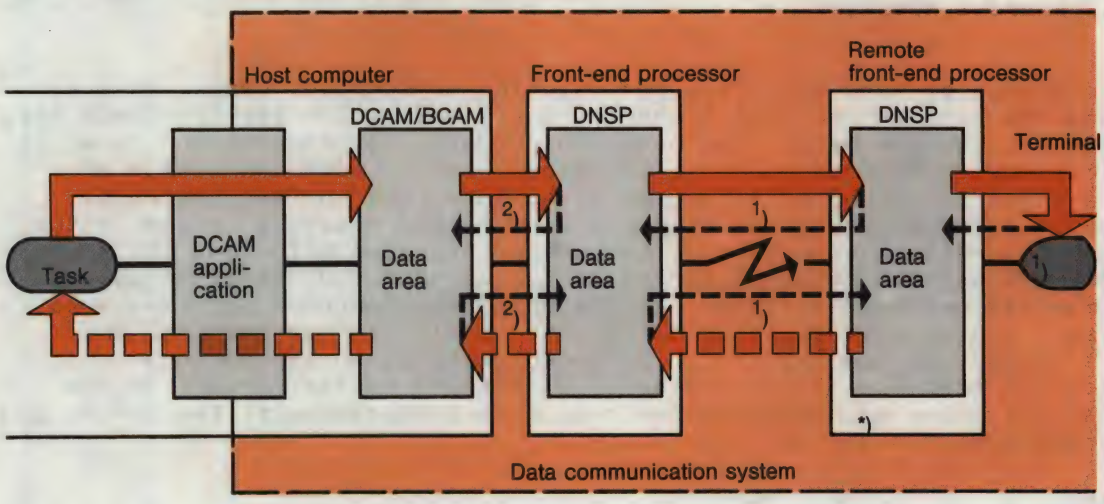
A DCAM application can be protected against unauthorized **connection of a task** within the host computer. If the application is to be non-shareable, the connection of a secondary task is not possible. In the case of a shareable application, unauthorized connection of a secondary task can be prevented by means of a password.

Predefined applications can already be protected against unauthorized opening (by a user password) in the resource definition file at the system generation stage. This RDF password must be provided by the primary and secondary task.

The unauthorized **setup of a virtual connection** can be prevented within the data communication system. Firstly, by DCAM applications not accepting connection requests, either permanently or for a certain period of time, and secondly, by requiring that a password be specified in a connection request. Finally the user can opt to accept or reject a connection request on the basis of the connection message and/or the address of the requesting partner.
These procedures ensure that erroneous or intentional illegal access can be monitored. Passwords can also be dynamically altered for this purpose (see 3.4).

2.2.3.2 Data Security

Data protection is a further feature of the security concept. On the physical data transmission level, **protocols with error recovery** are used (HDLC, MSV1, MSV2, LSV1, NEA) guaranteeing the integrity of the transmitted data by means of suitable error detection and automatic error correction facilities (cf. Fig. 2-7).



Transmission of data (message with sequence number)

1) Positive acknowledgment of data communication procedure (ACK)

2) Positive acknowledgment of message I/O

Transport acknowledgment

*) The transport acknowledgment is generated here when the message has arrived at the destination (teminal) i.e. has been acknowledged positively.

Fig. 2-7 Example of message acknowledgment

The **allocation and distribution of the storage capacities** to virtual connections rule out mutual interference (e.g. by memory monopolization) and introduce a further security factor. In addition to this, the user can request **transport acknowledgments** for transmitted messages. He can opt for negative and positive, only negative or no acknowledgments. A negative acknowledgment is generated when a message cannot be delivered to its destination, regardless of the reason for this. A positive acknowledgment is generated when the message arrives at the destination (i.e. entered in the receiver's data area). The originator, which added a sequence number to the message, receives the acknowledgment, which was generated at the destination and transported through the network, under this number.

The originator can receive the transport acknowledgment like any other message. However, he can also receive the transport acknowledgment as an asynchronous notification (see 4.1.5 as well). In a task group, there is the further possibility of always transferring the transport acknowledgment to the primary task or to the task requesting the acknowledgment (see also 3.3.2).

Restriction:

COBOL users cannot receive transport acknowledgments as asynchronous notifications.

2.2.3.3 Express Messages

Another aspect of data security, the solution of conflict cases, is also provided for. Unhindered data transmission is ensured in the TRANSDATA data communication system by connection-oriented data flow control and capacity distribution. The existing buffer areas are used by each connection only being allowed to occupy a certain subarea. Thus, the buffers cannot be completely occupied by the data of one connection. However, to ensure that a connection remains operable in the case of blockage affecting it, short, **fixed-length** express messages can be delivered to the destination as **high-priority messages** which bypass the data flow control. Such messages "overtake" the others, so to speak. If the destination so desires, the express message is transferred to it immediately as an asynchronous notification (cf. 4.1.5), otherwise it is entered as far forward in the queue as possible. Express messages cannot be transmitted if the connection is operating with a virtual terminal.

Restriction:

COBOL users cannot receive express messages asynchronous notifications.

2.2.3.4 Data Flow Control

The system can reduce the load on an overloaded connection by temporarily interrupting the transmission of messages. This can affect both normal and express messages.
As soon as the connection is ready for use once more, the user can be informed, by means of a GO signal, that the jam has been cleared and that transmission of messages may be recommenced.

2.3 PLANNING PROGRAM STRUCTURE

When a DCAM program is being planned, its function may already be defined; however, it is also possible that, due to the facilities offered, the problem will be re-formulated or even formulated for the first time. The following functions are typical of those handled by a DCAM program.

2.3.1 Functions of a DCAM Program

DCAM programs are created in order to implement transaction processing in BS2000. As opposed to timesharing, its object and implementation are completed when the transaction task is started.
Moreover, DCAM programs usually solve problems which can be solved with specific BS2000 facilities (see Table 2-1)

Transaction mode functions	Supported in BS2000 by:
Dialog between a few individual terminals and a program	- Synchronous processing of DCAM calls - Management of primary tasks (non-shareable DCAM applications)
Dialogs in individual steps or in a sequence of inquiries and responses	- Queues for access to specific or any partners - Accompanying information on the connection for transfer with the message (containing, for example, the address of a memory area) Restriction: cannot be used in COBOL.
Processing of inquiries from a large number of terminals by one program system	- Asynchronous processing of DCAM calls with P1 EVENTING AND CONTINGENCIES - Management of primary and secondary tasks (shareable DCAM application) - Support of reentrant programming
Data transfer from programs in the same or in other host computers of the TRANSDATA data communication system	- Synchronous or asynchronous processing of DCAM calls - Event-controlled processing with P1 EVENTING AND CONTINGENCIES
Forwarding of data to exclusive data processing programs in the same host computer and coordination of the cooperation	- Use of local or common memory areas with COMMON MEMORY and serialization - Coordination by means of P1 eventing, inter-task communication or user switches
Formatting of messages for line-by-line output or forms display on CRT screens	- Support by means of conversion from virtual terminals to physical characteristics of individual terminal types (VTSU)
Utilization of specific terminal characteristics	- Possibility of "physical" programming of terminals

Transaction Mode Functions	Supported in BS2000 by:
Processing of inquiries in the shortest possible time to provide the terimnal user with good response times	- Task control in timesharing mode - Rapid file access with user PAM - Reentrant control of programs with shared code in conjunction with good memory usage and low paging rate
Processing of different requests from a terminal in one program system	- Message distribution via distribution code oriented queues
Data security	- Transmission of transport acknowledgments for messages - Transmission of express messages - Prevention of multiple file access to single records: ISAM shared update or blocks (half pages): user PAM shared update
Protection against unauthorized access to programs or program systems or to files	- Password protection of the DCAM application - Password protection of the virtual connection - Passwords for the protection of files against unauthorized reading, writing and processing

Table 2-1: Transaction processing with BS2000

The programmer has at his disposal:

the services of the TRANSDATA DCM (Data Communication Methods) interfaces, of the Control System and of the data management system DVS.

It is expedient to subdivide DCAM programs. In the case of shared-code programming, a subdivision into operation code, constants area and dynamic work area is required. This also facilitates documentation and subsequent maintenance.

In the case of **simple functions**, such as the operation of a small number of terminals, single-step interactive communication, no special security requirements etc., programs can be subdivided into a communication part and a data processing part.

More **complex functions** require extensive interaction of the various interfaces and therefore only allow demarcation in accordance with the specific problem. It is beyond the scope of this manual to set forth any general rules on this subject. By providing for program systems which control one primary task and several secondary tasks, the system supports the structuring of complex functions. It is also possible to subdivide the coding into ASSEMBLER and COBOL modules.

Before going into the specific functions of a primary task and of the secondary tasks in detail, it seems expedient to list the essential elements for the creation of a DCAM program.

2.3.2 Basic Structure of a DCAM Program

Two preparatory stages have to be completed in a DCAM program or program system before actual data transmission can begin:

1st stage: **Opening a DCAM application.**

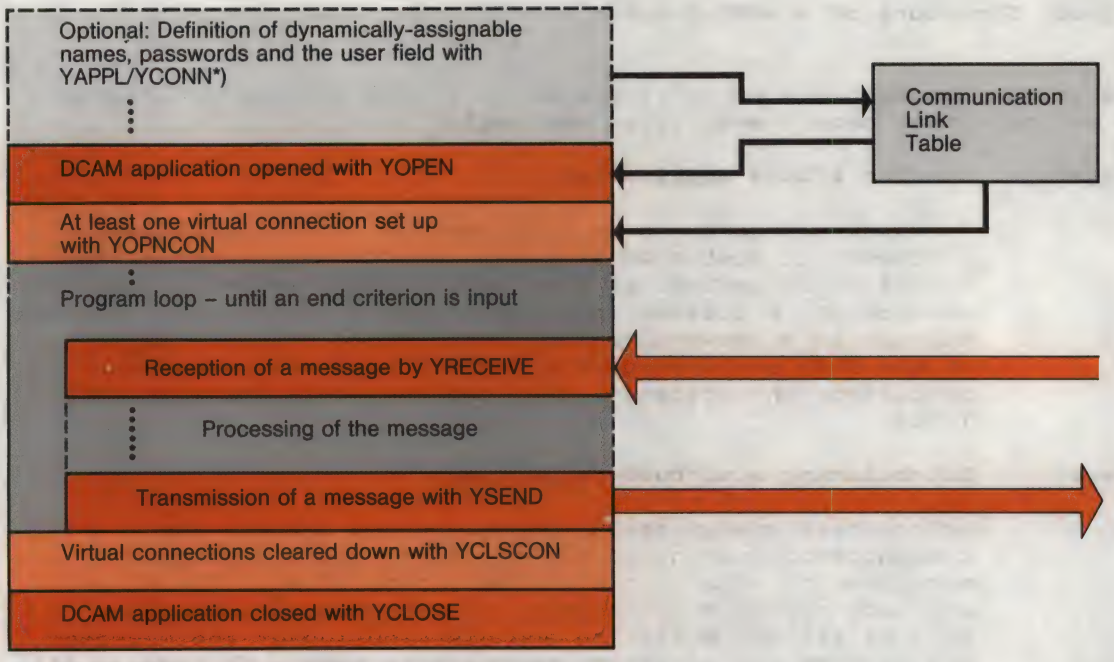
To allow a program to be addressed by other communication partners, it must open at least one DCAM application. The name that is defined at that point forms, together with the name of the processor (processor node), the symbolic network address in a network region. Furthermore characteristics of the DCAM application are defined, and passwords are specified. The statement that executes this stage is called YOPEN.

2nd stage: **Establishing a virtual connection.**

Before data transmission can take place between two communication partners, one of the partners must send a request to establish a virtual connection to the other, who must accept this request, i.e. a virtual connection must be set up. Buffering and distribution of the messages are carried out according to the specifications made at this point. Moreover, it is necessary to specify what kind of message editing is desired or whether the user will take care of message editing himself. In addition, the communication partners can exchange accompanying information. The statement for this stage is YOPNCON ACCEPT (accept connection request) or YOPNCON ACQUIRE (output connection request).

Following the two preparatory stages, messages can be **received** with YRECEIVE and **transmitted** with YSEND.

After completion of data transmission, a virtual connection is either closed down explicitly by the program with YCLSCON or implicitly by either the closing of the DCAM application by means of the YCLOSE statement or the termination of the program. In addition, a virtual connection can also be cleared down from the terminal by inputting an agreed end criterion which must initiate the execution of YCLSCON in the program. Abnormal termination of a virtual connection due to disconnection of the terminal, line failure, etc., can be reported to the program.



*) in ASSEMBLER only; with COBOL the corresponding command mode statements should be used

Fig. 2-8 Schematic layout of a DCAM program

Dynamic name assignment can also influence the structure of the program. The dynamically allocated names, passwords and the user field must be entered in the CLT (Communication Link Table) prior to the opening of the DCAM application or connection setup. A separate program or a leader should therefore contain the YAPPL or YCONN calls or commands /APPLICATION and /CONNECTION should be used accordingly (cf. Fig. 2-8).

Restriction:

The YAPPL and YCONN calls are not available for COBOL programs.

After each call issued to DCAM, a check must be made to see whether it was properly executed. **Feedback information (FDBK)** is provided for this purpose. The result of the check may be that a call is repeated or has to be executed in a different way, that other measures have to be taken, or that the program run has to be terminated. The error recovery routine can immediately follow the call but, as it is usually the same or similar measures that have to be taken, it will be coded only once and used repeatedly. In this case, a common error recovery routine would have to be created.

2.3.3 Control of Primary and Secondary Tasks

A primary task, i.e. the first task to open a DCAM application, contains all the essential DCAM functions (see preceding section). In a task group this task is automatically the **controlling task** which contains the connection function, distribution code control, and password definitions, in short everything that concerns the task group as a whole.

At YOPEN time, VERIFY can be used to check that the task is chronologically the first to open the application. The primary task can transmit and receive messages just as the secondary tasks. When the DCAM application is closed in the primary task, it is also closed for the secondary tasks.

The structure of the program controlling the primary task is dependent on the answers to various questions which are listed in Table 2-2.

Questions concerning the structure of a program controlling a primary task	Measures taken if the answer to the questions is positive
- Is the configuration to be operated small? - Is the mean time between the arrival of two messages considerable (low workload)? - Is the job so limited in scope that subdivision into different processing modules is not necessary?	The primary task should do all the processing; secondary tasks are not required and not permitted (non-shareable DCAM application)
- Is the mean time between the arrival of two messages short? - Is the configuration large? Is a short processing time required?	The primary task and the secondary tasks should be controlled by a shared-code program. As the primary task must carry out specific functions, this should be provided for in the program. Or alternatively, the program should be structured on a modular basis and only specific modules be utilized and loaded by the tasks
- Do the inquiries arriving via virtual connections require different types of processing? - Do the distribution codes in the message change? - Are passwords to be used to control the connection of a task to a DCAM application for security reasons? - Is all data to be checked? - Are additional control functions to be implemented?	The primary task should control distribution code allocation to the secondary tasks and thus message distribution. It will define a password for the connection of secondary tasks. It will check messages that cannot be delivered to a destination within the group. The primary task will have further means at its disposal for the control of the task group within the framework of common memory, P1 eventing, etc.

Table 2-2: Control of a primary task

A **secondary task**, i.e. a task which opens a DCAM application but is not the first to do so, has no influence on the characteristics of the DCAM application, and may have to specify a password in order to join the group. It can use VERIFY to check whether it opened the application later than the primary task. It can neither establish nor close down virtual connections. It has no control function within the group. Its function is to **transmit** and **receive** messages and to process them. Therefore, the only aspect of significance with regard to the structure of the program is the way in which YSEND, YRECEIVE or YSENDREC calls are issued and processed. Attention must be paid to any controlling actions by the primary task (connection control, distribution code allocation of incoming messages, etc.). If the DCAM application is closed in the secondary task, this has no effect on the other tasks of the group.

2.3.4 Access to Terminals

An important question in the planning of DCAM programs is how the terminals are to be accessed.

The DCAM programmer has a choice of two methods:

- physical programming
- use of virtual terminals.

Physical programming requires the operation of a terminal with the control characters it understands (see User's Guides for individual terminals). The programmer wins considerable flexibility, but has also the inconvenience of operating with control characters. The virtual terminals relieve him of this inconvenience because all essential functions can be implemented with these standardized terminals (but not with locally connected terminals).

The **virtual terminals** are user service software modules in the TRANSDATA data communication system. At the logon stage, the communication partners agree on which virtual terminal can or should be used. Certain aspects have to be clarified in order to do this, such as whether a hardcopy unit is connected to a display terminal or a light pen is to be used.

The programmer has a choice of two virtual terminals: the line terminal and the form terminal.

2.3.4.1 Line Terminal

The line terminal is implemented in the system by the VTSU (Virtual terminal support) module. The user only has to order the data into **logical lines** with a device-independent "new line" control character. Conversion of these lines into the actual lines of the terminal is performed by the system. During this procedure, logical lines may be subdivided into further lines if the available line length of the device (typewriter terminal, data display terminal, printer terminal) is not sufficient.

When a program receives messages from a line terminal, the end of the lines is indicated by the same device-independent "new line" character. During input and output, the beginning of a message is also the beginning of a line and the end of the message the end of the line.
The line terminal only accepts printable characters.

Fig. 2-9 shows the message path through the system when the line terminal is used.

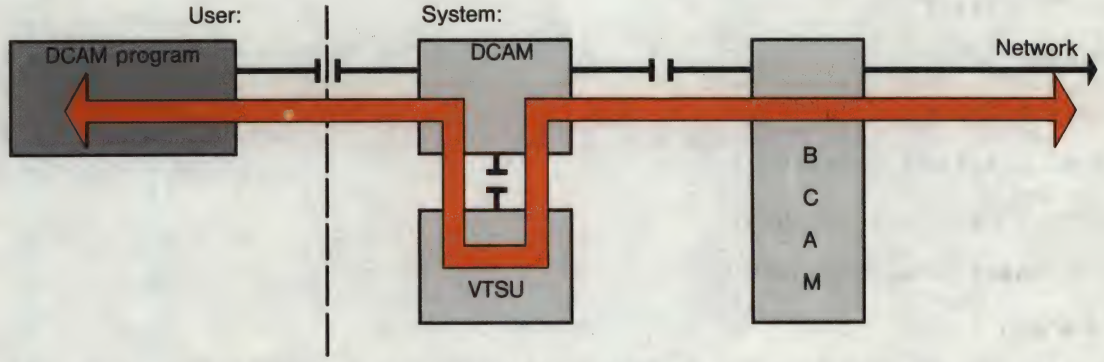


Fig. 2-9 Transmitting and receiving data to be represented in line format

The following logical control characters are also available to the user:

- new page
- mark-in
- mark-out
- shift-in
- shift-out

as well as

- output function key code
- structuring of output.

Additional record control character:

- logical end-of-line character

Logical display control characters:

- marked display 2
- marked display 3
- marked display 4
- blanking
- start proportional spacing
- end proportional spacing
- start unprotected area
- start protected area
- end protected area

Further logical control characters:

- delete character
- backspace
- current position
- vertical position (line)
- horizontal position (column)
- control sheet draw-in/eject
- left margin
- line spacing
- character spacing
- maximum line length
- maximum number of lines
- substitute character
- subscript line
- superscript line

Physical control characters:

- ESCAPE
- transparent transfer of first character
- horizontal tabulator
- vertical tabulator

The control character 'new page' in the output text of a message results in:

- output of a logical end of line
- page throw, i.e.

for data display terminals: The screen is cleared
Positioning to start of screen (not with 8161 using roll-up)

for printer terminals: Page throw

for typewriter terminals: Paper advance (10 lines)

The control character 'mark-in' causes the text which follows to be output in italic script or in flashing characters. This applies until the control character 'mark-out', 'new line' or 'new page' is encountered, or until the end of the message concerned.

The control character 'shift-out' switches to the second character set, if one is available. The user can subsequently switch back to the standard character set by means of 'shift-in'.

The function 'output of function key code' enables the user to have the device independent code of function keys or short messages output as the first byte of a message.

The function 'structuring of message' enables the user to specify whether the whole message, or each logical line of the message, is to be treated as a structural entity. There follows a brief explanation of the two structural types, using the 816x Data Display Terminals as an example.

Taking the message as the structural unit means:

The message length is restricted by the system output buffer. By modifying one character the whole message can be transmitted back again.

Taking the logical line as the structural unit means:
The message length is unrestricted, provided the logical lines are no longer than 255 characters. Logical lines can be modified individually, and thus transmitted back selectively.

The 'hardcopy' function enables the user to have the output on the screen logged simultaneously on a hardcopy unit.

The control character 'logical end of line' resets a special terminal display to standard display (standard, first character set, unprotected). The defined end-of-line character is output.

The control character 'start of protected area' outputs subsequent text characters as protected text on the screen, i.e. they cannot be overwritten or transmitted to the system.

The control character 'end of protected area' causes subsequent text characters to be output unprotected to the terminal.

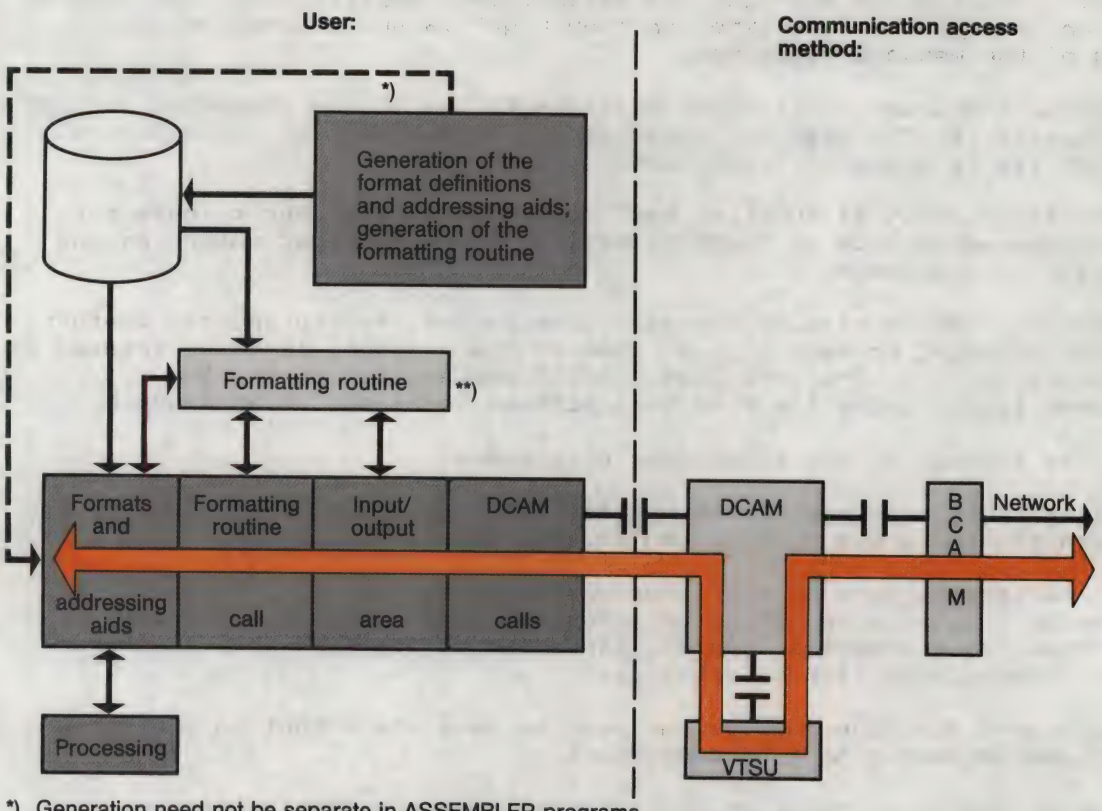
The control character 'delete character' causes a text character to be deleted from the output text and not to be transmitted to the terminal.

The control character 'backspace' causes the next text character to be superimposed on the preceding character. This enables a character which is not in the character set to be displayed.

2.3.4.2 Form Terminal

The form terminal requires the use of the terminal mapping support modules and macros.

The forms are described with the macros of the FORM interface. In COBOL programs, this must be done separately; the forms are later linked into the COBOL program as separate modules. Symbolic description maps are provided in ASSEMBLER and COBOL for the processing of the form fields. The program cooperates with a mapping routine for the transmission and reception of messages. The forms are edited by this routine for output on the physical terminals (typewriter terminal, data display terminal). Messages from the terminal are also converted by this routine (cf. Fig. 2-10).



*) Generation need not be separate in ASSEMBLER programs.
**) The formatting routine is linked into the user program.

Fig. 2-10 Transmitting and receiving formatted data

If desired, the following additional functions can be executed for the line terminal and for the form terminal:

- upper case/lower case processing
- processing of backspace characters
- simultaneous output on hardcopy unit
- sounding an audible alarm

2.3.5 Implementation of Distributed Processing

The TRANSDATA Communication System allows communication between:

- terminals and applications
- applications and applications

Here, both terminals and applications can form part of an extensive network, comprising host computers, front-end processors, remote front-end processors and terminal computers.

Apart from the connection to terminals, this opens up the following possibilities for a DCAM application:

The partner may be

- a DCAM application
- a UTM application
- a PDN application

2.3.5.1 A DCAM Application as a Partner

The DCAM application as a communication partner is described in chapter 3. The existence function of a DCAM application is described in section 3.1. Establishing a virtual connection between DCAM applications is described in section 3.2, while data transmission between DCAM applications is explained in section 3.3.

2.3.5.2 A UTM Application as a Partner

Communication between a DCAM application and a UTM application is possible if the DCAM has been generated as a communication partner for the UTM application and if a virtual connection has been established between the two applications. This is true whether the applications are in the same or in different host computers.

There are three different ways of establishing a connection between a DCAM and a UTM application:

- The UTM application has been generated so that when it is started a connection is established. The connection is set up if the DCAM application exists at this moment and explicitly accepts the connection request.
- The UTM application has been generated so that connection requests from the DCAM application are accepted. The connection is established if both applications exist and if the DCAM application has issued an explicit connection request.
- A UTM administration command can be used to cause a connection request to be issued to a DCAM application.

When transmitting data care must be taken that the UTM application has been provided with a structure. The UTM application is composed of subroutines, which can be addressed via the transaction codes which they have been allocated. When a transaction is started the data must be included at the beginning of the transaction code.

2.3.5.3 A PDN Application as a Partner

A PDN application is generated in the operating system of a terminal computer using the KOGS generator language. The PDN application comes into existence with the arrival of a message for the PDN application:

- from a connected terminal
- from an existing PDN application in the same computer,
- and with the arrival of a connection request from another communication partner in the TRANSDATA network.

PDN applications can also establish virtual connections themselves.

When establishing connections and transmitting data, it should be noted that DCAM handles a PDN application as an application. It follows accordingly that the system does not carry out message handling. In particular, the user cannot employ any virtual terminals.

3 DCAM FUNCTIONS

The functions provided by DCAM can be subdivided into 4 groups:

- Existence function
- Connection function
- Data transmission function
- Name assignment function

These functions are described in detail in the following. The layout of this chapter corresponds to that of chapter 3 of the ASSEMBLER Programmer's Guide (see "DCAM Macros") and the COBOL Programmer's Guide (see "DCAM COBOL Calls"), where the corresponding calls are listed along with the required operands.

3.1 EXISTENCE FUNCTION

The existence function of the DCAM interface performs the following operations:

- **Opening or generation** of a DCAM application **(YOPEN)**.
  This is the basic function of DCAM, and is executed in different ways depending on the type of DCAM application involved.
- **Changing the status (YSETLOG)** of a DCAM application.
  This function can be used to dynamically change the state of the DCAM application, i.e. its readiness to process connection requests.
- **Testing the status (YINQUIRE)** of a DCAM application.
  As dynamic changing (see preceding item) is possible, this inquiry function is also provided.
- **Closure** of a DCAM application **(YCLOSE)**.
  The DCAM application is closed automatically at the end of the program; this function can close it at any point in time during the program run.

3.1.1 Opening a DCAM Application

A DCAM application is generated when the first program opens it. At the same time the application is defined as shareable or non-shareable. A non-shareable application can only be opened by one program and is also closed by that program. A shareable application can also be opened by other programs (secondary tasks). These, however, only link up with the application and have no influence on the definitions made by the first program to open/generate the application (primary task). It is necessary to specify whether the distribution code oriented queues are to be used for message distribution. This results in different variants of the YOPEN call.

Specifications for dynamic name assignment are dealt with in a separate section (see 3.4).

3.1.1.1 Non-Shareable DCAM Application

A DCAM application which is to be non-shareable (NSHARE) can be opened by only one task. A second attempt to open this application will be rejected. Within the bounds of this restriction, it is expedient to lay down further definitions, which may vary depending on a number of factors (see Table 3-1).

Requirements	Definitions
The name of the DCAM application is not to be generated by the communication access method.	Name of the DCAM application
Only the DCAM application is to issue connection requests (NLOGON). In this case, the partners must either be terminals or DCAM applications accepting connection requests.	Attribute NLOGON: Connection requests will not be accepted.
It is not known in the program whether and when a partner wishes to log on or the partner is not known.	Attribute LOGON: Connection requests will be accepted.
The application is protected by a password in the RDF (APPPW=, see the manual Generation of a Data Communication System).	Password USEPW
Unauthorized connection requests are to be rejected automatically.	Password (LOGPASS), which has to be specified by the communication partners.
Asynchronous notifications are to be accepted.	Reference to the identifier of a contingency routine generated when an ENACO call was issued (see (8)). When a notification arrives, this routine is activated. Note: unnecessary for COBOL
Certain new functions belonging to a DCAM version from 8.0 onwards are to be used.	Specification of DCAM version number (DCAMVER).

Table 3-1: Definition of non-shareable DCAM applications

Restriction:

When COBOL is being used, the name of the DCAM application must always be specified by the user.

3.1.1.2 Primary Opening of a Shareable DCAM Application

A shareable DCAM application can be opened by more than one task. The first task to open it is the primary task. Unless otherwise specified, messages are not distributed via distribution codes (SHARE, NDISCO), i.e. the originator-oriented or common-receiver queue(s) are used. The name of the DCAM application must be defined in the program. For further definitions, which are dependent on various factors, see Table 3-2.

Requirements	Definition
The task is to be the first opening task as it is to fulfill the functions of a primary task. If it is not the primary task, the YOPEN is to be rejected.	Check with VERIFY = PRIMARY.
It can be taken for granted that the task is the first opening task, or all programs linking to the application have the same program logic.	No check, i.e. VERIFY = NO.
The possibility of unauthorized connection of secondary tasks cannot be excluded and is therefore to be prevented.	Password USEPASS, which must be specified by the secondary tasks for subsequent opening. Note: If the DCAM application is already protected by a password (APPPW=, see manual "Generation of a Data Communication System") in the RDF, the USEPASS password must be identical to that APPPW= password. The USEPW operand is then also analyzed in the primary task; it must contain the RDF password.
Only the DCAM application is to issue connection requests (NLOGON). In this case the partners must either be terminals or DCAM applications accepting connection requests.	Attribute NLOGON: Connection requests will not be accepted.
It is not known in the program whether and when a partner wishes to log on, or the partner is not known.	Attribute LOGON: Connection requests will be accepted.
Unauthorized connection requests are to be rejected automatically.	Password (LOGPASS), which must be specified by the communication partners. Note: LOGPASS and USEPASS cannot be changed in an existing DCAM application.

Requirements	Definition
Transport acknowledgments for this application are as a rule processed in the primary task.	PRIMTASK attribute for the transfer of transport acknowledgments.
Transport acknowledgments are processed by the task that sent the message with the corresponding sequence number (SEQNO).	REQTASK attribute for the transfer of transport acknowledgments.
This application does not process any transport acknowledgments because user security procedures are to be implemented.	NOTACK attribute: positive or negative transport acknowledgments are not transferred.
New functions belonging to a DCAM version from 8.0 onwards are to be used.	Specification of the DCAM version number (DCAMVER).
Asynchronous notifications are to be accepted.	Reference to the identifier of a contingency routine generated when an ENACO call was issued. When a notification arrives, this routine is activated. Note: not necessary for COBOL

Table 3-2: Description of shareable DCAM applications opened by the primary task

3.1.1.3 Primary Opening - Use of Distribution Codes

The primary task that opens a DCAM application can specify that a distribution code is to be used (SHARE,DISCO) in place of the default option for message distribution. This is especially useful if the programs involved perform different jobs but are accessed by the same partners. All the tasks involved must then define distribution code names permitting association of distribution code(s) and tasks (DISNAME). For further definitions, see Table 3-2.

3.1.1.4 Secondary Opening

A task which is not the first to open a DCAM application must comply with the definitions laid down in the primary task. Consequently, it must use the defined name of the DCAM application and also set ATTR=SHARE (shareable DCAM application). The default option provides for message distribution via the originator-oriented or common-receiver queues. Further definitions are listed in Table 3-3.

Requirements	Definitions
The primary task has defined a password to guard against unauthorized access or the application is protected by an RDF password.	Password (USEPW) for linking up with the DCAM application as specified in the primary task or RDF.
The task must always be a secondary task as this is stipulated in the program logic; if necessary, YOPEN is to be rejected.	Check with VERIFY=SECONDARY.
It can be taken for granted that the task is not the first to open the application, or, alternatively, the program logic checks this requirement.	No check, thus VERIFY=NO.
Asynchronous notifications are to be accepted. Restriction: There are no asynchronous notifications intended for the secondary task at the COBOL interface.	Reference to the identifier of a contingency routine generated when an ENACO call was issued. When the notification arrives, this routine is activated.
The primary task specified a DCAM version number on opening the application.	Specification of the same DCAM version number as in the primary task (DCAMVER)

Table 3-3: Description of shareable DCAM applications opened by a secondary task

3.1.1.5 Secondary Opening - Use of Distribution Codes

During secondary opening of a DCAM application, the same message distribution method must be used as during primary opening by the primary task, i.e. the distribution method is uniform within a task group. Distribution codes are used in this variant which means that the SHARE and DISCO attributes are specified in the primary task. The same DCAM application name and the SHARE attribute must be specified in the secondary task. Furthermore, a name must be defined (DISNAME) to be used for distribution code allocation. All further definitions are listed in Table 3-3.

3.1.2 Altering the State of a DCAM Application

The various states of a DCAM application are shown in Fig. 3-1.

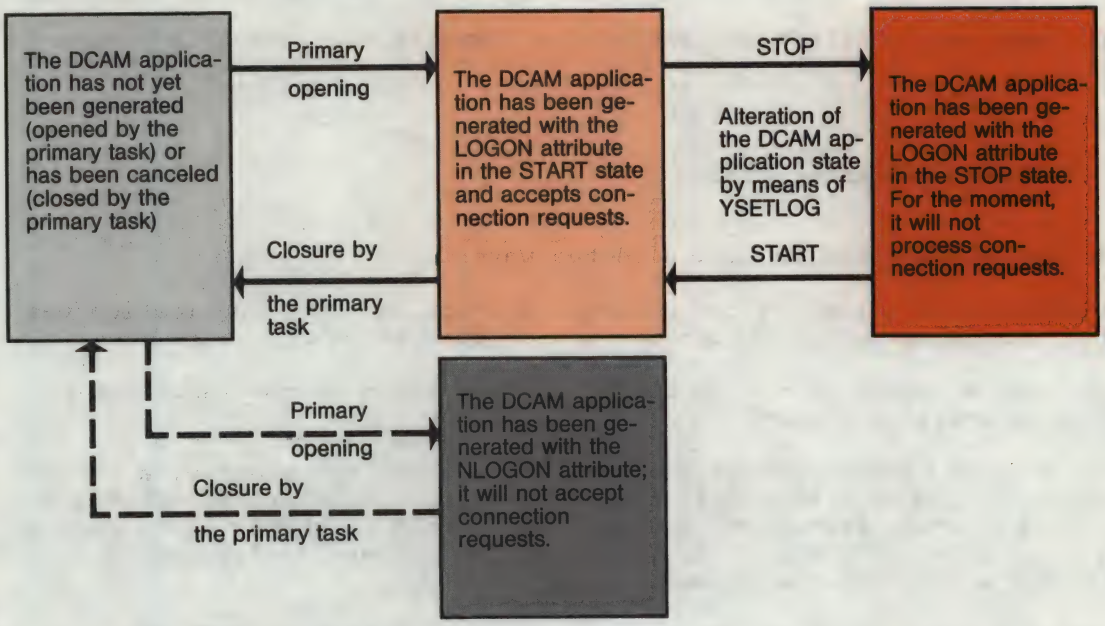


Fig. 3-1 DCAM application states

A DCAM application can be designed not to accept connection requests (NLOGON attribute). This attribute cannot be altered later for an existing DCAM application.

If, on the other hand, such requests are to be accepted (LOGON attribute), this can be temporarily prevented.

Connection requests are reported to the DCAM application, or, if several arrive, posted in a queue, until the primary task notifies by means of the YSETLOG call that it no longer wishes to accept such notifications (STOP), possibly because the maximum number of permissible connections has been reached or because it wishes to issue its own requests for some time. If conditions change, the primary task can at any time reestablish the original state, i.e. requests will no longer be rejected but will be accepted by the primary task or posted in the queue (START).

3.1.3 Testing the State of a DCAM Application

With the YINQUIRE call, the user can test the state of the DCAM application that he himself has opened and also the state of a DCAM application which was opened in the same host computer and whose name he knows.

He should use the "APPSTAT" variant of the YINQUIRE call for this purpose.

3.1.4 Closing a DCAM Application

The DCAM application can be closed in two ways:

- implicitly by terminating the program in which the DCAM application was opened (normal or abnormal termination or task abortion)
- explicitly by means of the YCLOSE call or by means of the /SHUTDOWN or /BCLOSE operator commands.

Explicit closing with YCLOSE will be necessary when, for example, a program is to process several DCAM applications consecutively or if definitions are to be changed during execution. The application can be opened again by the primary task with new definitions (name, attributes etc.) after being closed. (See 3.4 for name assignment function.)

The **secondary task** can close the application at any time without any consequences for the other tasks in the group.

Closure by the **primary task** results in:

- the DCAM application being canceled;
- secondary tasks being informed through initiation of the COMEND contingency routine (see 4.1.5) or by feedback information for an incomplete call or the next call;
- all existing connections being cleared down.

This also means that data which has already arrived but has not yet been accepted is no longer accessible. Pending connection requests are deleted.

3.2 CONNECTION FUNCTION

The requirement for data transmission is the setting up of a virtual connection between the communication partners. This is the second preparatory step after a DCAM application has been opened.

The connection function performs the following operations:

- **Virtual connection setup (YOPNCON).**

  connection requests can be output and incoming requests can be accepted.

- **Interrogation of entries on partners and connections (YINQUIRE).**

  As the characteristics of the connection must be defined when the connection is established, this function serves to interrogate transmitted proposals, etc.

- **Rejection of a connection request (YREJLOG).**

  If there are reasons for not establishing a certain connection after a LOGON notification, this function can be used.

- **Changing the characteristics of a connection (YCHANGE).**

  This call was created in order to provide the capability of changing certain characteristics of a connection during the program run without having to close the connection and then open it again.

- **Deletion of a request (YCLSCON)**

- **Closing of a connection (YCLSCON)**

  This enables selective closing of a connection

3.2.1 Establishing a Virtual Connection

A **standard procedure** was designed for connection setup. Depending on the type of computer using this procedure, different means are used to implement the various procedural steps. The calls required for a DCAM application are shown in Figs. 3-2 to 3-7. The terminal operator procedure is described in Appendix A.3.

Two steps must be taken before a connection can be established:

- One partner must send a connection request to another.
- The other partner must accept the request.

The coordination of these steps is performed by the system:

- **The requests** are delivered to the partner and, if others are pending, entered in a queue.

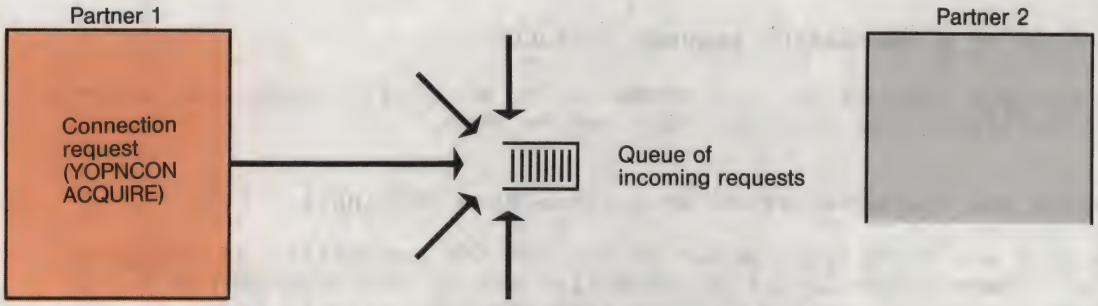


Fig. 3-2 Connection request

- It is possible to have the arrival of a request indicated by an **asynchronous notification** (LOGON notification see 4.1.5).
- After the notification has been issued, a **contingency routine** can establish which partner made the request. Following this, the request can be accepted or rejected (Fig. 3-3).

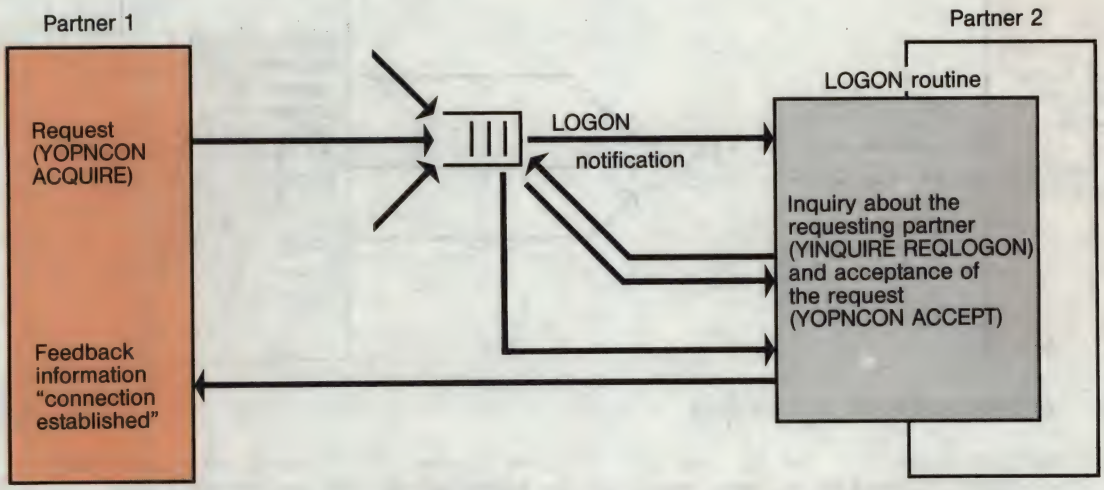


Fig. 3-3 LOGON notification; request testing and acceptance

- The partner can **interrogate** this queue **without** having received **a notification** (see 3.2.1.3 and 4) to ascertain whether entries are present and can, depending on the result, issue accept calls for this request. (YOPNCON ACCEPT SPEC, Fig. 3-4)
- He can also reject the request or, by not accepting it, cause the request to be deleted after a period of time defined in the system (/BCTIMES command, see "Network Management in BS2000").

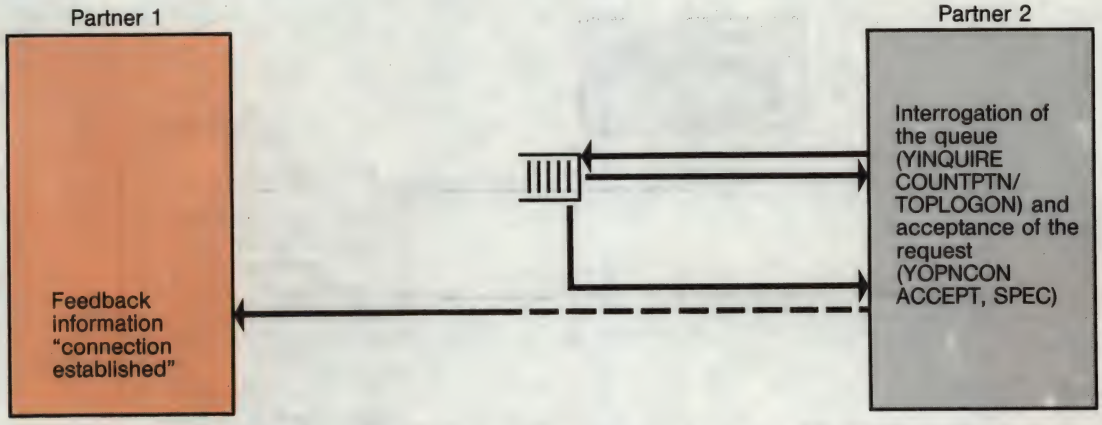


Fig. 3-4 Partner information inquiry and request acceptance

- He can also issue accept calls **without inquiring.** If the request arrives within the specified period of time, it is accepted. A number of this type of acceptances are entered in a queue if the requests have not yet arrived. They may be entered for a specific (SPEC) or an arbitrary (ANY) partner (Fig. 3-5).

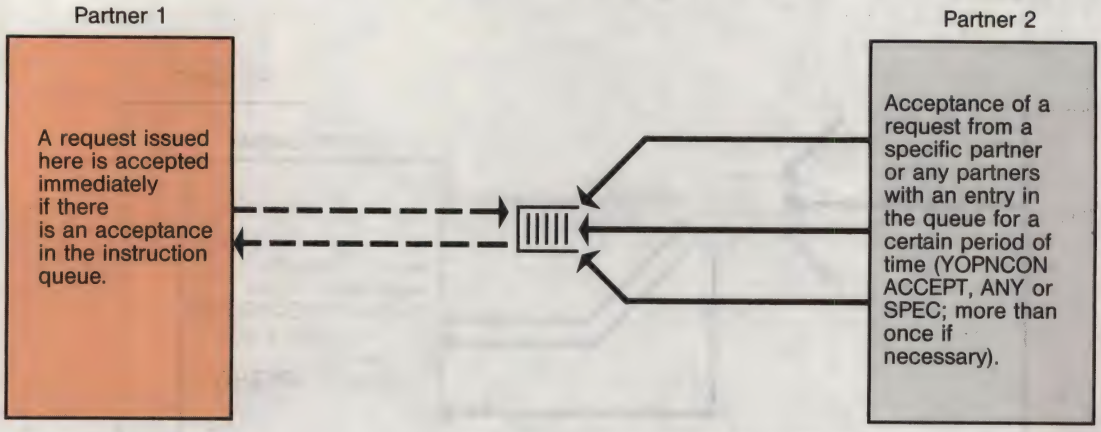


Fig. 3-5 Acceptance of a request

- A further possibility is to assign partner names to a DCAM application **when the communication access method is generated and started** (see XSTAT macro in "Generation of a Data Communication System"). When the DCAM application is opened, or a /BCIN command referring to the processor node of the proposed partner, the named partners are proposed for connection setup by means of the PROCON notification. As a result of the notification, a request must be issued (YOPNCON ACQUIRE). The connection is only established if the proposed partner accepts this request (Fig. 3-6).

  Restriction:

  PROCON notifications are not issued to COBOL programs.

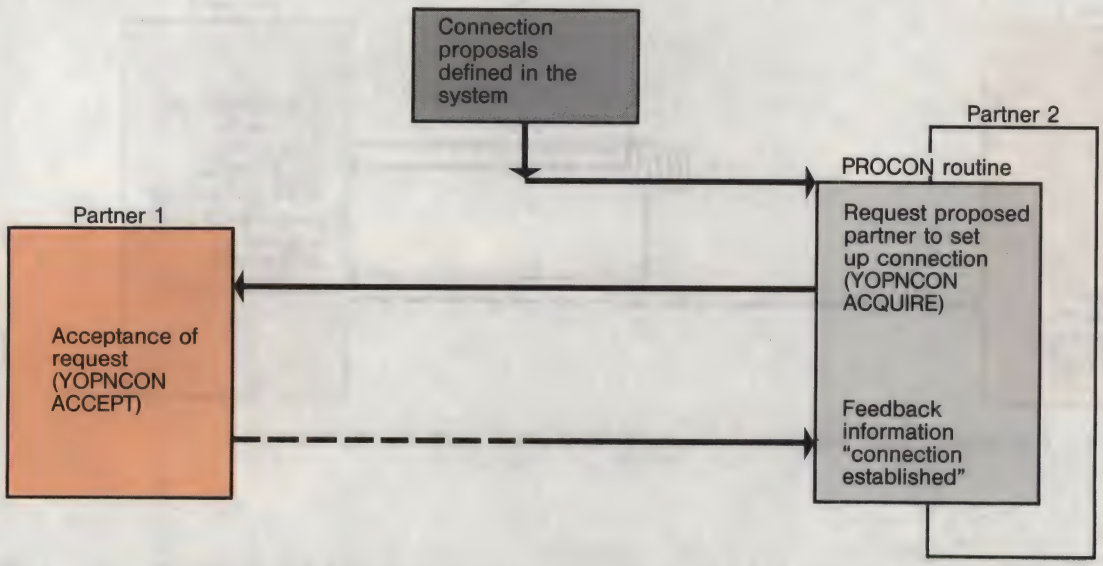


Fig. 3-6 PROCON notification request to proposed partner and acceptance

- When generating and starting the communication access method, **connections** (XKON macro), as well as connection proposals, may be predefined.

  These connections are already established whenever the communication access method is started. However, data transmission is not possible until the primary task issues a call which is usually used as a connection request (YOPNCON ACQUIRE).

  If the partner is a terminal, it is sufficient for this to be activated and to be operational.

Every communication partner that transmits a logon call has to define the desired connection characteristics. When a connection request is made, the partner is informed of that part of the connection characteristics which affects him. This part consists of:

- **The use of a data communication protocol (DEPROT);**

  Restriction:

  Not supported by the system in DCAM V8.0.

- **The type of message editing used (EDIT);**

- **Specification of the partner initiating data communication (PROC).**

Fig. 3-7 shows how this information is requested and defined:

Partner 1 sends partner 2 a connection request (YOPNCON ACQUIRE), and also makes suggestions for EDIT and PROC to partner 2. Partner 2 can react to the call from partner 1 in two ways:

- By requesting the suggestions from partner 1 (YINQUIRE) and then checking and either accepting or rejecting them, i.e. by sending other values for EDIT and PROC back to partner 1 (YOPNCON ACCEPT).

- By sending its own values for EDIT and PROC to partner 1 without considering the suggestions.

In each case the values sent by partner 2 are binding for partner 1. This means the values suggested by partner 1 for EDIT and PROC need not necessarily correspond to the current values, and therefore the user must inform himself of the current values after the connection is established.

Both in the case of a request for connection setup (YOPNCON ACQUIRE) and the acceptance of the connection (YOPNCON ACCEPT) the partners can send each other connection messages.

If a connection was predefined, its characteristics have been defined during system generation.

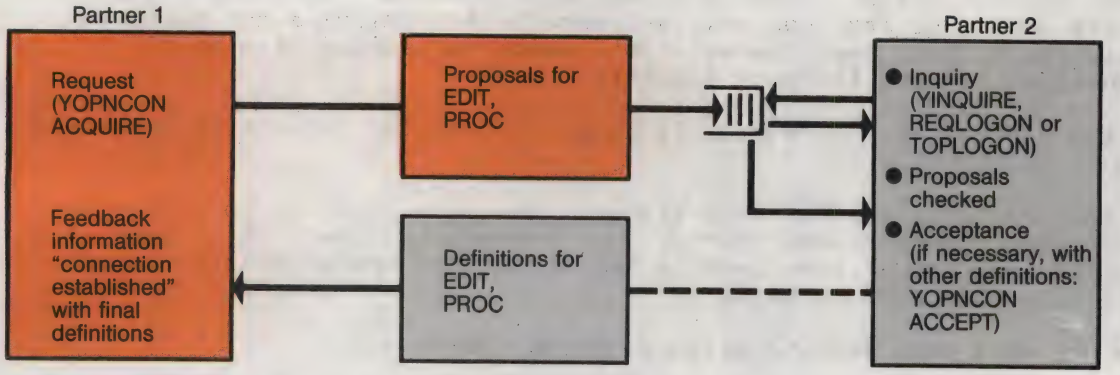


Fig. 3-7 Agreement on characteristics

3.2.1.1 Definition of the Connection to be Established

The connection to be established has to be defined in the DCAM program. The following information is required:

- **Name and processor name (= address) of the partner**

  This information is used to address a partner when a request is issued or a request is accepted by a specific partner (SPEC).
  If an accept call was issued for any partner (ANY), DCAM returns the partner and processor names after completion of the call.

- **Accompanying information**

  Here, the user defines an optional character string which, for example, is added to every message he receives via this connection. It may be the address of a data area to be assigned to this connection or the address of a routine which is to be activated specifically for this connection. This method is particularly useful for access to the common-receiver queue with RECEIVE ANY, but also for distribution code oriented when several tasks are using the same distribution code name (see also 3.2.1.4). The content of the accompanying information is optional; however, it must not exceed 4 bytes.

  Restriction:

  Accompanying information cannot be defined in COBOL.

- **Handling of excess-length messages**

  Depending on the job to be done, a choice must be made between two possibilities; see Table 3-4 for details.
  This entry can be changed later on during the execution of a receive call (see 3.3.2).

Requirements	Definitions
- Only messages of fixed length are expected on this connection. - A message that is longer than expected must be errored. The excess-length part is to be truncated.	The value TRUNC is specified for PROC. The excess-length portion of messages that are longer than expected is truncated. The excess-length condition is indicated but the excess-length part is ignored.
- It is not possible to predict the length of the messages arriving over this connection. The length assumed to be most common is therefore specified and an input area reserved for it. - The possible excess-length is not to be discarded; it will be fetched with a further receive call after having been indicated.	The value KEEP is specified for PROC. The excess-length condition is indicated when the 1st section is received, and the excess-length part is saved for a further receive call.

Table 3-4: Handling of excess-length messages

- **Message code**

  As different codes can be used on the transmission line sections in a data communication system, it is left to the user to select a code according to the job definition at hand; Table 3-5 provides an overview of the options.

Requirements	Definitions
- The user is working with virtual terminals. - He wishes to receive messages in the code of his own host computer (as a rule in EBCDIC). - He always transmits messages in the code of his own host computer.	The value SYSCODE is specified for PROC. If necessary, the messages are converted in the data communication system.
- The user is not working with virtual terminals. - He wishes to receive messages in the code used by the communication partner. - He transmits messages in the code which the partner can process.	The value BINARY is used for PROC. The messages are transmitted in any code (transparent).

Table 3-5: Message code

- Data flow control
  It is possible for a connection to become overloaded, in which circumstances no notifications can be handled.
  By issuing a GO signal, the user can ascertain when the connection becomes free once more, permitting the transmission of data (SIGNAL).

- The **Logon password** is specified when a connection request is to be made. It must be specified in the same way as the partner (in this case a DCAM application) defined it during the opening procedure (see 3.1.1.1).

- Where **messages are distributed by means of distribution codes,** the reference to the detailed description of the distribution codes must be specified here. This is described in detail in section 3.2.1.4 in its further context.

- The maximum length of data (MAXLN) which is to be transmitted via this connection. This is a value ensuring optimum usage of the buffers provided by the system; it is not passed on to the communication partner.

- **Message characteristics which are agreed with the partner**

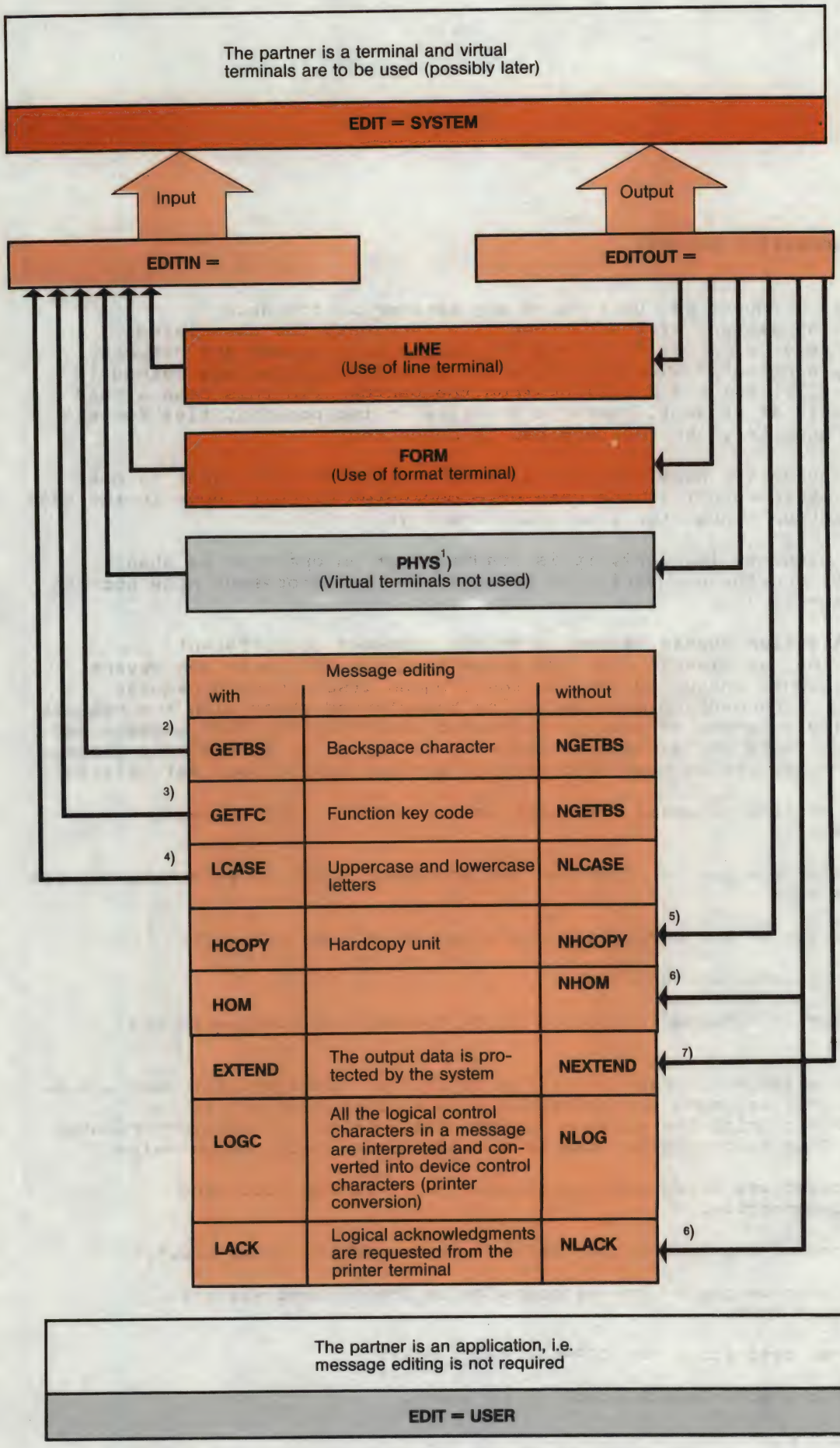
  As mentioned in the preceding section, the two partners must agree on some characteristics. These are also described here.

  Initiation of data transmission by one partner or the other is also defined. If the DCAM application is to initiate transmission, APPSTART is set. Otherwise, there is no definition (ANYSTART).

  The type of **message editing** used is defined. An overview of the options is provided in Fig. 3-8 (cf. also 2.3.4).

Note:

For connections with EDIT=SYSTEM, where ATTR=DISCO the distribution code is always edited in line mode.



1) Message headers, if present, are transferred in device code and the messages themselves in EBCDIC; a message may be divided into submessages (see Table 3-7 in 3.3.1).

2) 8103 Printer Terminals only
3) With EDITIN = LINE only
4) With EDITOUT = LINE/FORM only
5) With EDITOUT = LINE/PHYS and 8151, 8152 or 816X devices only
6) With EDITOUT = LINE only
7) With EDITOUT = LINE and 975X or 816X devices only

Fig. 3-8 Message editing options

3.2.1.2 Connection Request

A connection request can be sent to **any partner** in the data communication system. If the partner is a terminal, the associated processor (e.g. a remote front-end processor) will answer the request. Otherwise, a response will come from the communication access method in BS2000 (see 3.1.1.1 and 3.1.2) or from the partner (in this case a DCAM application). At present, there is a choice of two possibilities for the processor node to which the **terminal** is connected:

- If the processor node receives a request for a terminal that is not ready (switched off in the case of a dedicated circuit, busy in the case of a dial-up line, etc.), it will reject it.
- If the terminal is ready, it is assumed that an operator is also present, and the connection is established (the processor node accepts the request).

The **communication access method** in BS2000 responds to different criteria. The overview in Fig. 3-9 shows that the DCAM user has several control options, including some in the program itself during request processing. A connection message can be transferred along with the request for checking purposes or just for isolated transmission. This message can have any contents and a maximum length of 80 bytes. It is the only message that is transmitted without a connection already having been established.

A partner wishing to **issue a request** has to specify the following information:

- Address of the partner (partner and processor name) to which the request is directed.
- Description of the connection characteristics (see 3.2.1.1).
- Logon password, if required.
- A connection message (optional) to be transmitted along with the request.
- Type of message distribution (if distribution codes are not used), i.e. whether the messages transmitted via this connection are to be distributed using the common-receiver queue or the originator-oriented queue. This specification can be altered during data transmission.

If the request was accepted, the partner receives the following **feedback information:**

- The accompanying information defined by the partner (see 3.2.1.1)
- The maximum length of the messages to be transmitted via this connection (MAXLN).
- The final definition for DEPROT, PROC, EDIT.
- Partner characteristics (type of partner).

The possibility of having the statement contained in this call processed asynchronously is dealt with in the description of the language features (see 4.1.3).

Restriction:

Connection messages are only delivered to DCAM applications. Terminals connected to TRANSDATA 960 Communication Computers cannot receive connection messages.

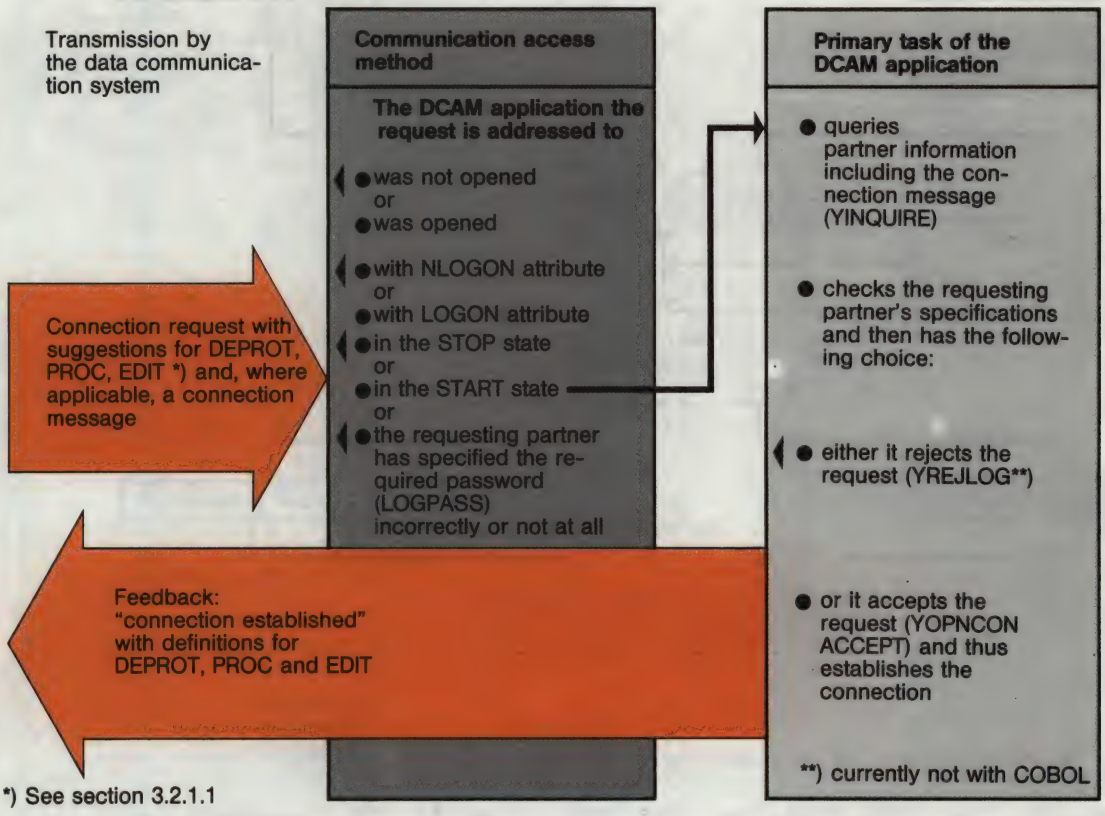


Fig. 3-9 Processing a request

3.2.1.3 Acceptance of a Request

The basic description of acceptance is contained in the preceding sections. This section contains a summary of the information which can be specified for acceptance. The inter-relationship of the various specifications is shown in Fig. 3-10.

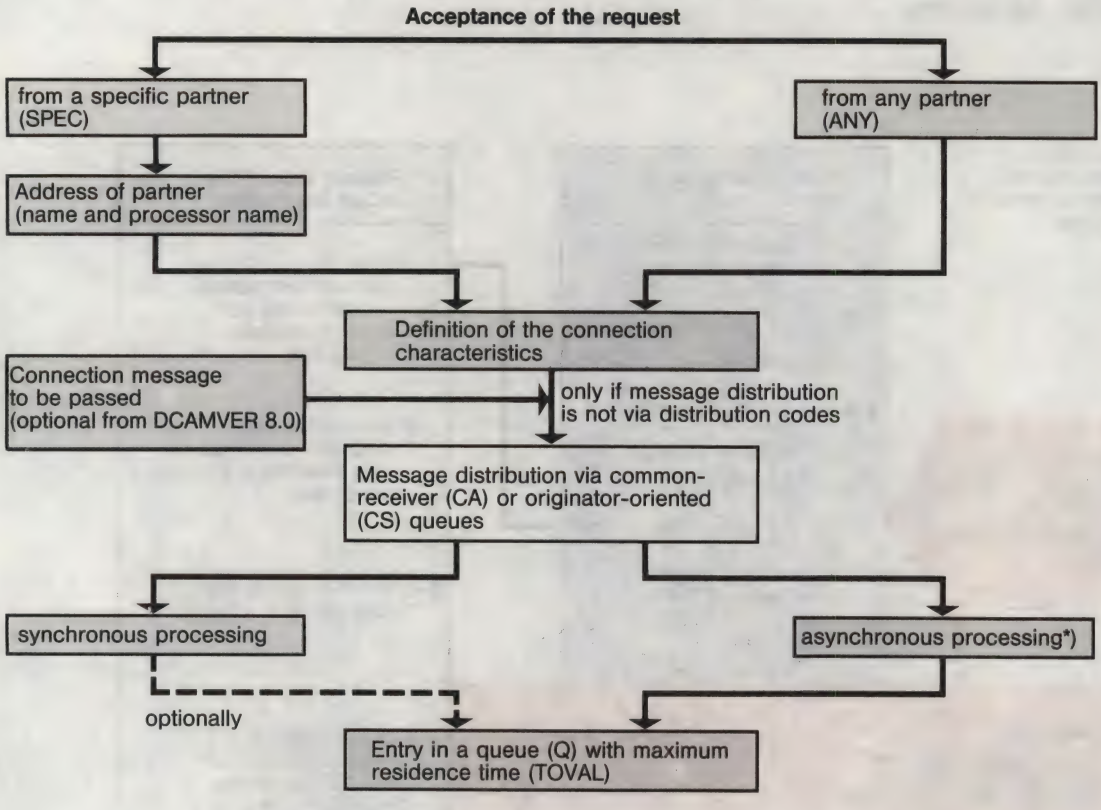


Fig. 3-10 Acceptance of a request

Once the call has been executed, the task receives the following information:

- the maximum message length on this connection (MAXLN) and
- some partner characteristics (essential features).

In addition, if the request accepted was that of any partner:

- the name of the partner
- the name of the partner's processor
- the accompanying information (cf. 3.2.1.1)

3.2.1.4 Connection Setup - Use of Distribution Codes

If messages are to be distributed by means of distribution codes (defined when the application is opened), a number of definitions must be made when the connection is set up.

As already mentioned (see 3.2.1.1), the description of a connection contains, in this case, a reference to that section that describes the distribution codes. Distribution codes are described in 2 stages (see Fig. 3-12) in order to provide maximum freedom in associating connections with distribution codes.

Stage 1 **Description of the code position in the message and of the Length** of the code(s) used. The code must be contained in the first 256 bytes of the message and can have a maximum length of 8 bytes.

In addition: reference to the description of the second stage, which may be present 8 times (COBOL) or 16 times (ASSEMBLER).

Stage 2 **Description of the codes** used by this connection. Up to 8 codes can be described.

These descriptions can be shared, i.e. several connections can use the same stage 1 description and several stage 1 descriptions can use a stage 2 description.

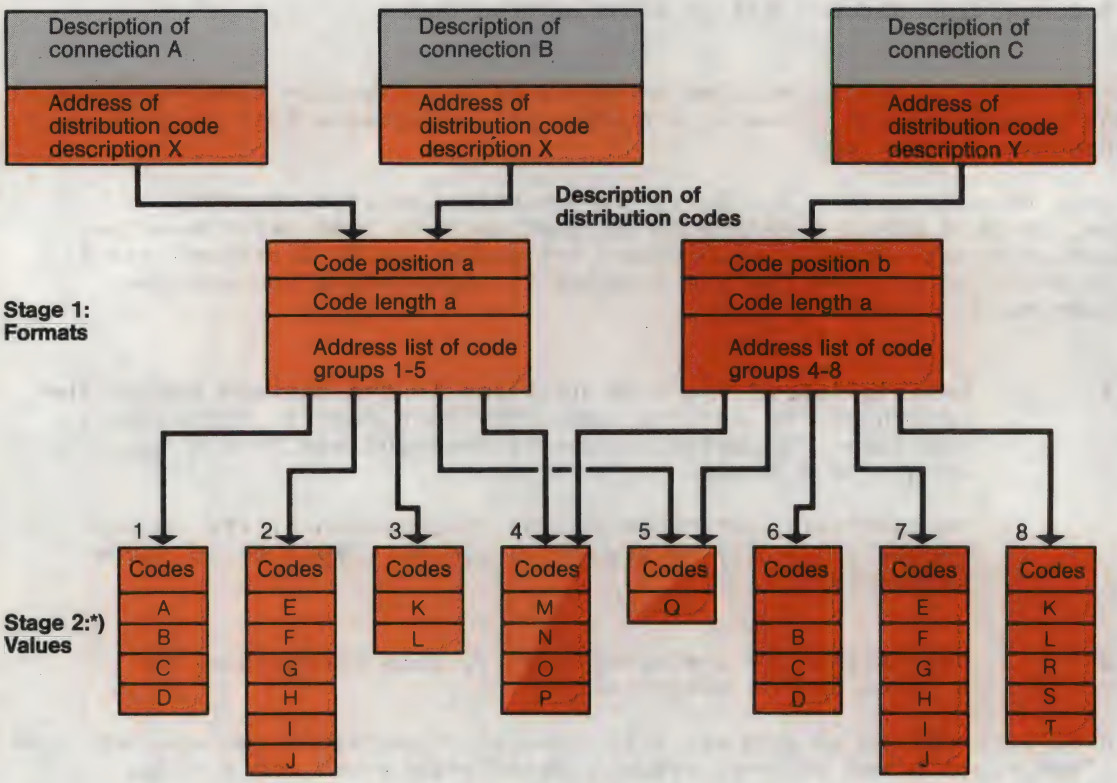
The following must be observed:

- The codes which are addressed by the same stage 1 description must be unique and
- must all be of the same length.
- If several stage 1 descriptions address a stage 2 description, the length specifications in the stage 1 descriptions must be identical.

The possibilities that result from this method of describing distribution code during the setting up of a connection are as follows:

- Code length, code position in the message and the code sign are defined when the connection is set up and cannot be altered.
- A partner can specify up to 64 (COBOL) or 128 (ASSEMBLER) different distribution codes in groups of 8. Thus, the partner can either reach one task via up to 8 different codes or he can read a varying number of tasks.

  The number varies because the assignment of codes and task via the distribution code name takes place during the execution. This assignment is controlled by the primary task by means of YPERMIT and YFORBID calls (see 3.3.5).



*) The codes of groups 1–5 and 4–8 must be unique in either case

Fig. 3-11 Description of the formats and values of distribution codes

3.2.1.5 Linking Up to a Predefined Connection

A predefined connection is determined during the **generation of the communication access method.** It is established when the communication access method is started. However, the two partners must first connect themselves to the predefined connection before data transmission can begin.

In the case of applications the primary tasks must make a connection request (YOPNCON ACQUIRE). In the case of a terminal this need only be activated and be operational.

Because the characteristics of a predefined connection are determined during the generation of the communication access method, and because the connection is automatically established when this is started, certain **special factors** are involved:

- If data is transmitted before the partner has linked up with the connection negative acknowledgments are given.
- If a partner wished to withdraw from a connection the LOSCON-ROUTINE is not initiated (the predefined connection remains established).

- A YOPNCON ACCEPT SPEC to a predefined partner can never be successfully terminated. If OPTCD=Q is specified, the call is entered in a queue and is terminated by timeout.

The following specifications cannot be made by a YOPNCON ACQUIRE if the connections are predefined:

- a connection message
- a LOGON password
- declarations on the data communication protocol (DEPROT)
- message editing by the system or the user (EDIT=SYSTEM USER)
- start data transmission (PROC=APPSTART ANYSTART).

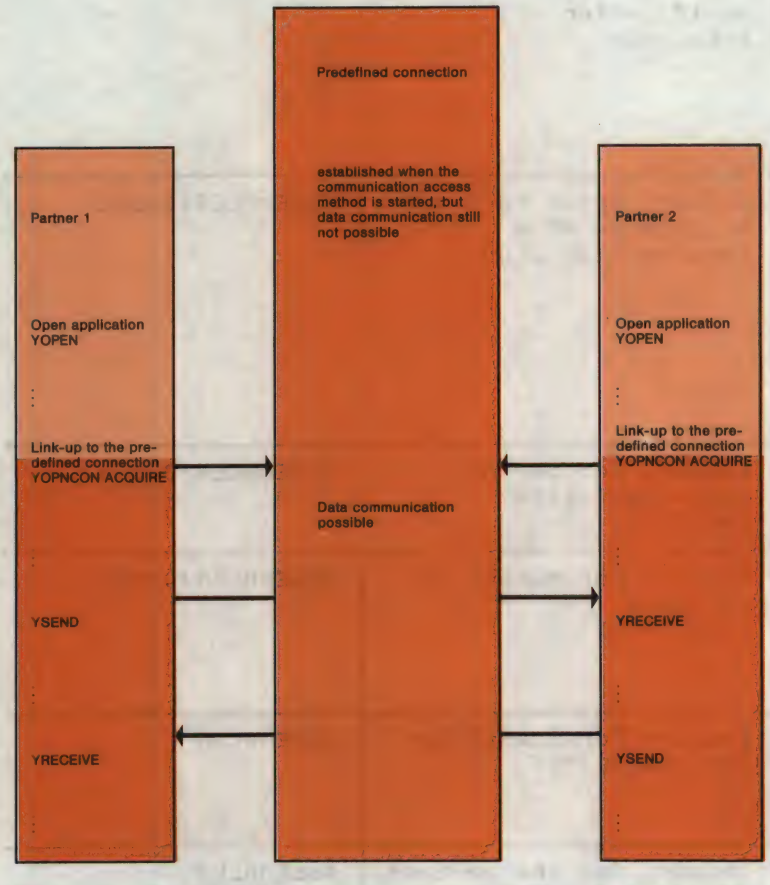


Fig. 3-12 Linking up to a predefined connection

3.2.2 Interrogation of the Entries on Partners and Connections

As already mentioned in the sections on connection requests and the acceptance of requests, it is necessary to test the entries on partners and connections. The YINQUIRE call, of which 6 variants are available, is used for this purpose. For secondary tasks, only the CIDXLATE, NAMXLATE and PTNCHAR variants are applicable.

Restriction:

In COBOL only 3 variants of the YINQUIRE call are available.

Table 3-6 shows the YINQUIRE variants and the programming languages in which each can be used.

Inquiry	YINQUIRE variant	Programming language
- Which partner is requesting connection? What are the proposed characteristics of the connection? - What connection message is to be transmitted and what is its length?	Inquiry after an asynchronous LOGON notification (REQLOGON)	ASSEMBLER
- Which partner is next in the queue of those requesting connection? - What are the proposed characteristics of the connection? - What connection message is to be transmitted and what is its length?	Inquiry prior to acceptance of a request (TOPLOGON)	ASSEMBLER/COBOL
- What characteristic feature does the partner have?	Inquiry for partner characteristics (PTNCHAR)	ASSEMBLER/COBOL
- How many partners have sent a request? - With how many partners has a connection been established?	Inquiry for number of partners (COUNTPTN)	ASSEMBLER/COBOL
- What is the identifier of the connection of which the partner and the processor names are known?	Inquiry for identifier CID (NAMXLATE)	ASSEMBLER
- What are the partner and processor names of a connection whose identifier is known?	Inquiry for the partner and processor names (CIDXLATE)	ASSEMBLER

Table 3-6: Inquiry variants

3.2.3 Rejection of a Connection Request

As described above (see 3.2 and Fig. 3-9), it is possible to receive a connection request via the asynchronous LOGON notification (see 4.1.5). It was also mentioned that information on the partner that sent the request can be obtained together with a connection message that may also have been transmitted (see 3.2 and 3.2.2). After all this has taken place and following a check on this information it is possible to decide not to set up a connection with the requesting partner and to reject the request. The YREJLOG call is provided for this purpose.

Only the address of the partner (partner and processor names) obtained with the YINQUIRE call need to be specified.

3.2.4 Changing the Characteristics of a Connection

The user can change some of the connection characteristics defined during connection establishment.

Only those characteristics can be changed which exclusively affect the partner making the changes, i.e. not those which have been agreed with the communication partner.

In the YCHANGE call, all of the values listed in the following must be set regardless of whether they are changed (new) values or the previously set (old) values.

The following can be changed:

- Accompanying information
- Handling of excess-length messages
- Code of the messages
- Values for EDITIN (Input message editing)
- Values for EDITOUT (Output message editing)
- If required, address of the distribution code description for changing distribution codes described in groups (the length and the position in the message and the code sign cannot be changed).

For further information on these values see 3.2.1.1.
(Definition of the connection to be established).

3.2.5 Deletion of a Request

A connection request sent to a partner can be deleted with the YCLSCON call.

The connection request is revoked if the connection is not yet established. If already established it is cleared down.

3.2.6 Closing a Connection

Explicit closing of a connection is possible with the YCLSCON call.

There is no obligation to do this, as all connections are implicitly closed when the DCAM application is closed. Uncollected messages are destroyed when the connection is closed. Therefore it is recommended that the final message be saved with a transport acknowledgment.

Explicit closing may be required for:

- time-limiting a connection
- regulation of data throughput (connections can be closed down in order to give other connections better throughput rates)
- freeing a terminal for another connection, etc.

3.3 DATA TRANSMISSION FUNCTION

The requirements for data transmission are fulfilled when a DCAM application has been opened and a connection established.

The data transmission function performs the following operations:

- **Transmitting a message (YSEND)**

  The message to be transmitted to a communication partner is transferred to the data area of the communication access system.

- **Receiving a message (YRECEIVE)**

  The message from a communication partner (any one, a specific one or one with a specific distribution code) is entered in the data area of the user program.

- **Combined transmitting and receiving (YSENDREC)**

- The data for a message is entered in the data area of the communication access method by means of a call, and the data of a message from the same communication partner is entered in the partner's own data area. This provides an effective means of interactive processing.

- **Deletion of receive calls and changing the CS/CA state of a connection (YRESET)**

  With this call a process can delete either all its waiting YRECEIVE calls from a specified application or a specified connection, and it can alter the CS/CA state.

The following facilities are available to control distribution by means of distribution codes:

- **Assignment of distribution code names to distribution code groups (YPERMIT)**

  The primary task assigns specific distribution codes to the distribution code name of one or more secondary tasks.

- **Cancellation of this assignment (YFORBID)**

  An assignment is canceled by the primary task.

3.3.1 Transmission of a Message

A message transmission call, which can be issued by any task of an application, results in the following action:

- The data to be sent to a partner in the form of a message is transferred to the data area of the communication access method.
- The access method is assigned the job of transmitting the data to the partner.
- The call is terminated and the access method starts transmission.

The access method receives a number of user specifications along with the message to be transmitted. They are supplemented by specifications defined when the virtual connection was set up (see 3.2.1.1).

Additional specifications required under particular conditions are listed in Table 3-7.

Requirements	Definitions
After logon, the system announced the maximum length on this connection.	The length of the message to be transmitted is specified within the maximum length limits.
- Acknowledgment reception was enabled when the application was opened (for the primary task=PRIMTASK or for the requesting task=REQTASK). - Transport acknowledgments are to be processed. - The YSEND call is to be immediately followed by a YCLSCON or YCLOSE call which means the connection may be closed before the data has been transmitted. The transport acknowledgment should always be waited for in this case.	A sequence number is specified so that the transport acknowledgment can be assigned to the correct message (maximum value=2047) when it arrives. A positive acknowledgment is requested (negative acknowledgments are generated automatically if the message cannot be delivered).
- Messages arriving on this connection are to be received via the originator-oriented queue as of now.	The originator-oriented queue is set (CS state of the connection).
- Messages arriving on this connection are to be received via the common-receiver queue as of now.	The common-receiver queue is set (CA state of the connection).
Message editing was not provided for by the system on this connection. The following was set: - EDIT=USER: The user edits the messages in such a fashion that the communication partner can process them. - EDIT=SYSTEM and EDITOUT=PHYS: The system performs blocking for devices with a small input buffer. Everything else is provided by the user in such a way that the partner can process it.	The message can be classified as follows: - ELEMENT: - SUBGROUP: - GROUP: This identification of the message is transferred by the network to the receiver, which receives it along with the message. If the receiver is a terminal, an element is transmitted as a submessage, while a subgroup or a group is transmitted as a message followed by transmission termination.
Message editing is provided by the system on this connection.	Special rules apply to the use of virtual terminals (see 2.3.4, 3.2.1.1 and Fig. 3-8).
A special situation has arisen (e.g. backlog on the connection), which was notified by the feedback information for a rejected YSEND. The partner is to be requested to issue receive calls immediately.	The message is declared as an express message, which means that its length may not exceed 8 bytes.
Upon opening of the connection PROC=SIGNAL was used to request the GO signal in the event of the connection being overloaded.	For each YSEND call a valid event identifier (EID) should be specified, via which a GO signal can be issued if necessary.

Table 3-7: Definitions for the transmission of a message

The message layout has to be varied in accordance with the **message editing** option selected:

- When a **Line terminal** is used, the user can subdivide the messages into logical lines. Each line is terminated with the "new line" character, hex 15. The text following the NL character is automatically placed at the beginning of the next line by the terminal. If the logical line length exceeds the line length of the data terminal, the logical line is subdivided automatically. The beginning of a message is automatically placed at the beginning of a line. Fig. 3-13 illustrates this by means of an example.

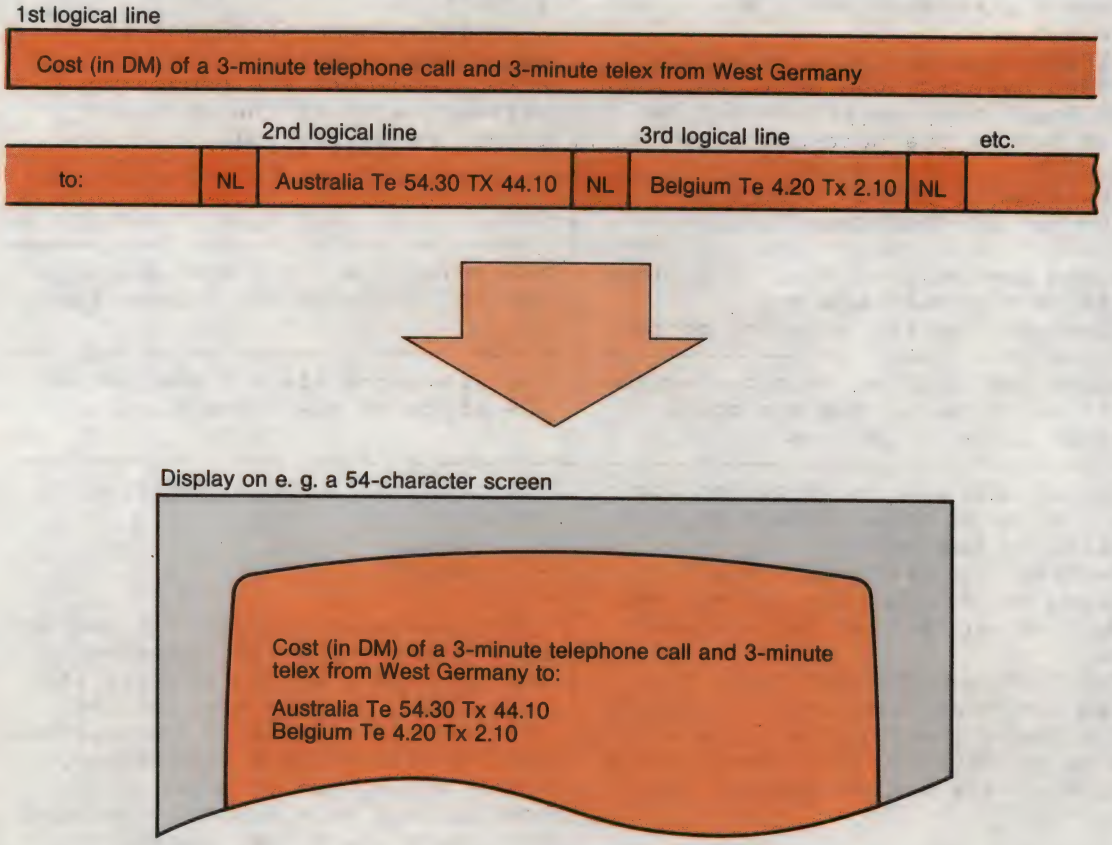


Fig. 3-13 Message to a line terminal

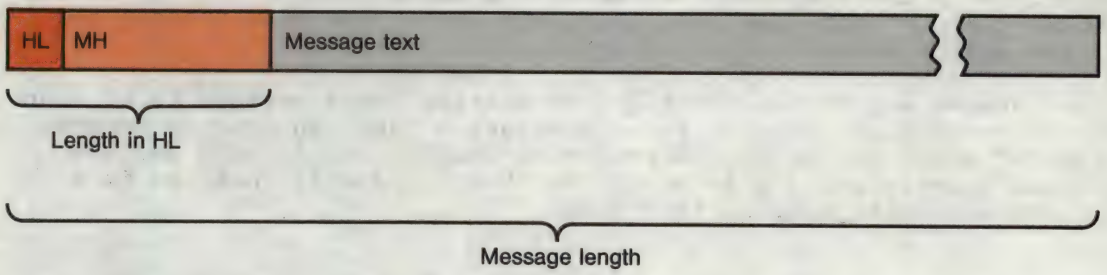
Additional functions available are:

Function	Control Character	Effect
- New Page	hexadecimal 0C	Page throw
- Mark-in	hexadecimal 1D	Output flashing or italicized
- Mark-out	hexadecimal 1F	Output normal
- Shift-out	hexadecimal 0E	Shift to second character set
- Shift-in	hexadecimal 0F	Shift to standard character set

- When a **format terminal** is used, the user transfers the message in the way he receives it from the mapping routine, but without the preceding record length field. The record length must be transferred from the record length field to the corresponding DCAM field.

- If the values **EDIT=SYSTEM** and **EDITOUT=PHYS** have been set for the connection, each message must be preceded by a header length byte. The binary number contained in this byte specifies of the following message header (inclusive of the length byte). The message header is transferred to the terminal without code conversion in contrast to the message proper, so that the user can use the device description during generation and analysis of the message header. For devices which do not process message headers, the message must nonetheless be preceded by a header length byte, and the length must be set to 1.

  Figure 3-14 illustrates how messages are to be formatted.



HL = Header length byte; contains length of HL + MH in binary

MH = Message header; length varies depending on the device function; in device code. Not required for devices controlled in other ways.

Fig. 3-14 Message layout for EDITOUT=PHYS

The message can also be subdivided into submessages in this case and also when EDIT=USER has been set (see Fig. 3-15). Each submessage must be appropriately marked.

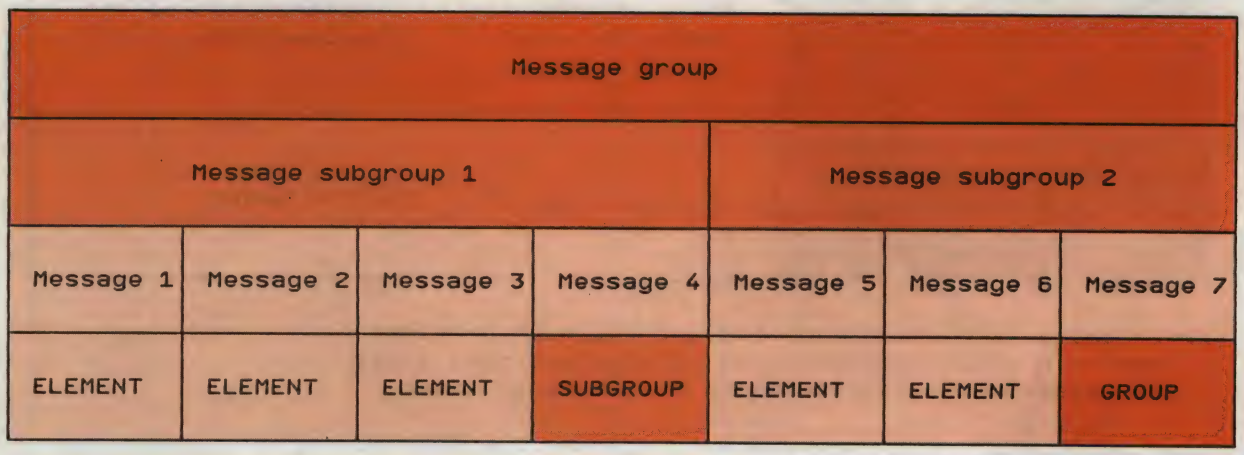


Fig. 3-15 Example of message structuring

Depending on the device type and status, different device responses have to be reckoned with (see "Terminal Access Support").

Note:

Shared physical lines (concentrator) are released each time a submessage has been transmitted.

3.3.2 Receiving a Message or Transport Acknowledgment

Reception of a message means that the message from a partner is entered in the data area of the receiving partner's program after having been removed from the queue.

This procedure is completed when the call is terminated provided that the call was executed synchronously. If the call was executed asynchronously, its completion merely means the communication access system has been assigned the job of transferring the message when it arrives. An agreed event identifier enables the user to receive a signal when the message has been transferred. The user can check for this signal and then process the message (see also 4.1.3).

This method allows any delays arising from waiting for a message to be used for other processing operations. Which messages a task can receive depends on the type of application, the message distribution method used and the type of queue selected, and also on whether the message in question is a normal or express message (see Table 3-8).

Type of message / Type of message distribution		Normal message	Express message
By means of distribution codes (DISCO)		Distributed to the task authorized to receive it according to distribution code 1)	Distributed to the primary task
Without distribution codes (NDISCO)	Via common-receiver queue (connection state CA)	Distributed to the first task to access the common-receiver queue (ANY)	Distributed to the primary task
	Via originator-oriented queue (connection state CS)	Distributed to the task that set the originator-oriented queue (caused the CS state)	

1) If the distribution code has not yet been provided for or has not been assigned to any secondary task, the message is distributed to the primary task.

Table 3-8: Message distribution and receiving

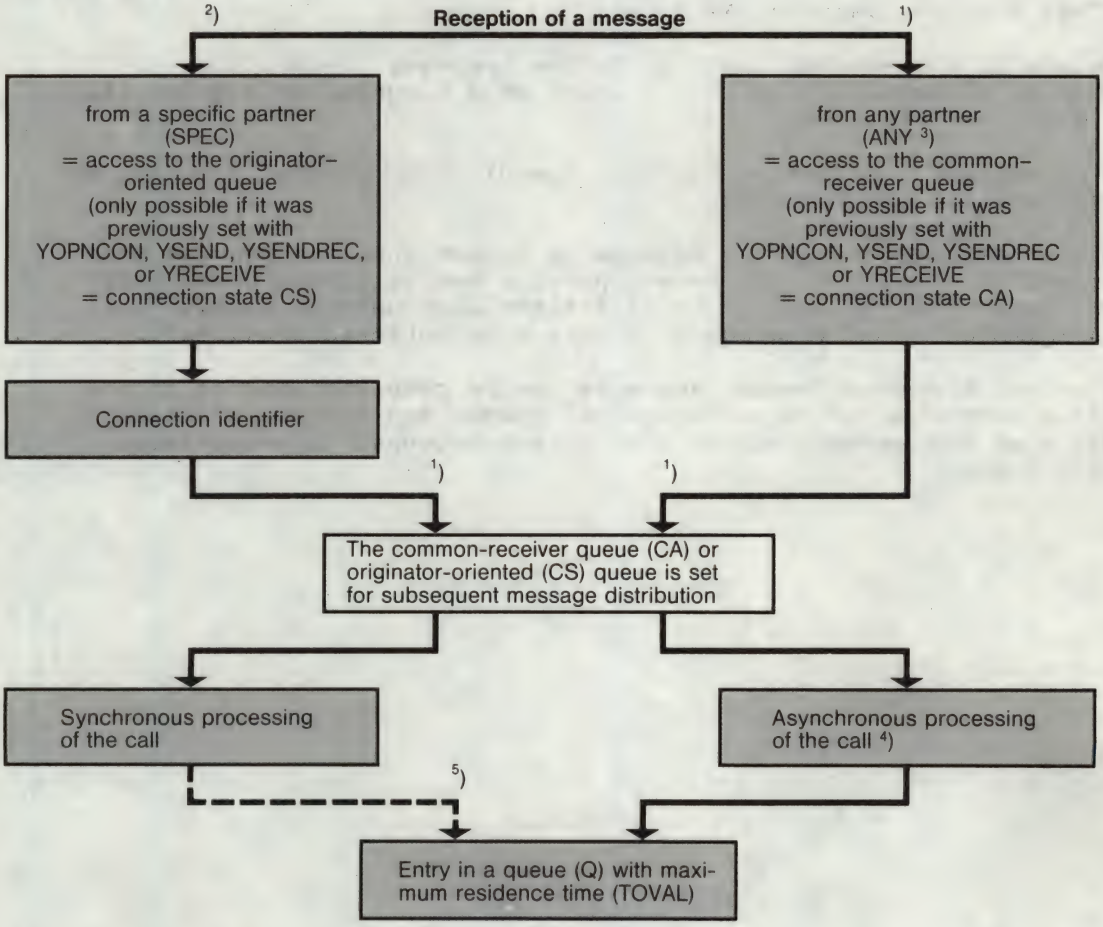
When issuing the YRECEIVE call, the user specifies further information beyond the connection description defined with YOPNCON (see 3.2.1.1).

- With respect to the **message**:

  he defines the address of the field in which the message is to be entered, along with the length of this field;
  he can change the handling of excess-length messages defined during connection establishment to match this message; i.e. he can redefine whether the excess portion of a message is to be kept so that it can be fetched later or whether it is to be discarded.

- With respect to **call execution**:

  interrelated information must be specified here, as illustrated in Fig. 3-16.



1) Only possible for message distribution without distribution codes
2) Setting of CS for this purpose not necessary in the case of message distribution with distribution codes
3) The call may be transmitted prior to connection setup. However, it is not executed until at least one connection has been established.
4) For further specifications, see section 4.1.3.
5) Optional: if not specified, the call is terminated immediately even if it cannot be executed.

Fig. 3-16 Specifications for receiving a message

After the message has been received in the specified area, further information is fed back by DCAM:

- **Originator specification,**

  if the message was received from any partner (ANY).

- **Sequence number** of message

  as specified by the partner or generated by the system (if a terminal was the originator).

- **Accompanying user information**

  as defined by the user when the connection was established (message received from any partner: ANY).

- **Structure** of the message (entries in the feedback field), i.e. whether an element, or the last element of a subgroup or a group (see Fig. 3-15) was received.

- **Type** of message (entries in feedback field), i.e. whether a normal or express message was received.

- **Length** of message or, if the message is longer than the storage area provided, the length of the excess portion not yet transferred (KEEP). If the excess-length part is to be fetched with another receive call, the originator-oriented queue must have been selected previously.

Whether or not **transport acknowledgments** can be received depends on the definitions made when the application was opened and on the information specified when the message which is to be acknowledged is transmitted (see Table 3-9).

Definition at YOPEN of the application in the primary task	Definition for YSEND or YSENDREC	
	A positive acknowledgment is requested (TACK)	A positive acknowledgment is not requested (NTACK)
The requesting task is to receive acknowledgments (REQTASK)	Transfer of positive and negative acknowledgments to the requesting task	Transfer of negative acknowledgments only to the primary task
The primary task is to receive all the acknowledgments (PRIMTASK)	Transfer of positive and negative acknowledgments to the primary task	
No acknowledgments are to be passed (NOTACK)	Neither positive nor negative acknowledgments are passed	

Table 3-9: Distribution of transport acknowledgments

Unless the use of asynchronous notifications was agreed (see 4.1.5), transport acknowledgments are received with the YRECEIVE call. All information on the type of acknowledgment etc. is provided in the feedback field of the call. The field 'TACKNO' also contains the sequence number of the message as defined during transmission. It is thus possible to identify the message.

It should be noted that the reception of a transport acknowledgment with YRECEIVE has no effect on the setting of the queue (CS/CA) even if this was specified in the call.

Restriction:

Where EDIT=SYSTEM, a second negative transport acknowledgment can be received when YSEND is specified. (This restriction applies through to DCAM V8.0).

3.3.3 Combined Transmitting and Receiving

The information given on the two separate calls also applies to the combined call. As the combination only permits access to the originator-oriented queue, the originator-oriented queue must have been set for this connection (=connection status CS) with YOPNCON or a previous YSEND or YRECEIVE.

Restriction:

The YSENDREC call is not available with COBOL.

3.3.4 Cancelling of Receive Macros and Changing the Connection State CS/CA

With this call asynchronous YRECEIVE macros can be canceled.

A **YRESET(ANY)** call cancels all ARECEIVE macros from any partner.

A **YRESET(SPEC)** call cancels YRECEIVE calls from a specified partner.

For applications with **ATTR=NSHARE or (SHARE, NDISCO)** a YRESET(SPEC) resets the connection state CS/CA at the same time.

3.3.5 Control of Distribution Code Assignment

The primary task controls distribution code is assignment by assigning a distribution code group to a distribution name or by deleting such an assignment. This is one of three necessary measures which may be taken in any order:

When setting up connections, the primary task defines the distribution code group for one or more connections (see 3.2.1.4).

The distribution code name is entered by one or more tasks when opening the application (see 3.1.1.3 and 3.1.1.5).

In this way the primary task assigns the distribution code name of one or more tasks to the distribution codes of one or more connections.

If the primary task deletes such an assignment, it inhibits data transfer from the connections concerned to the task concerned.

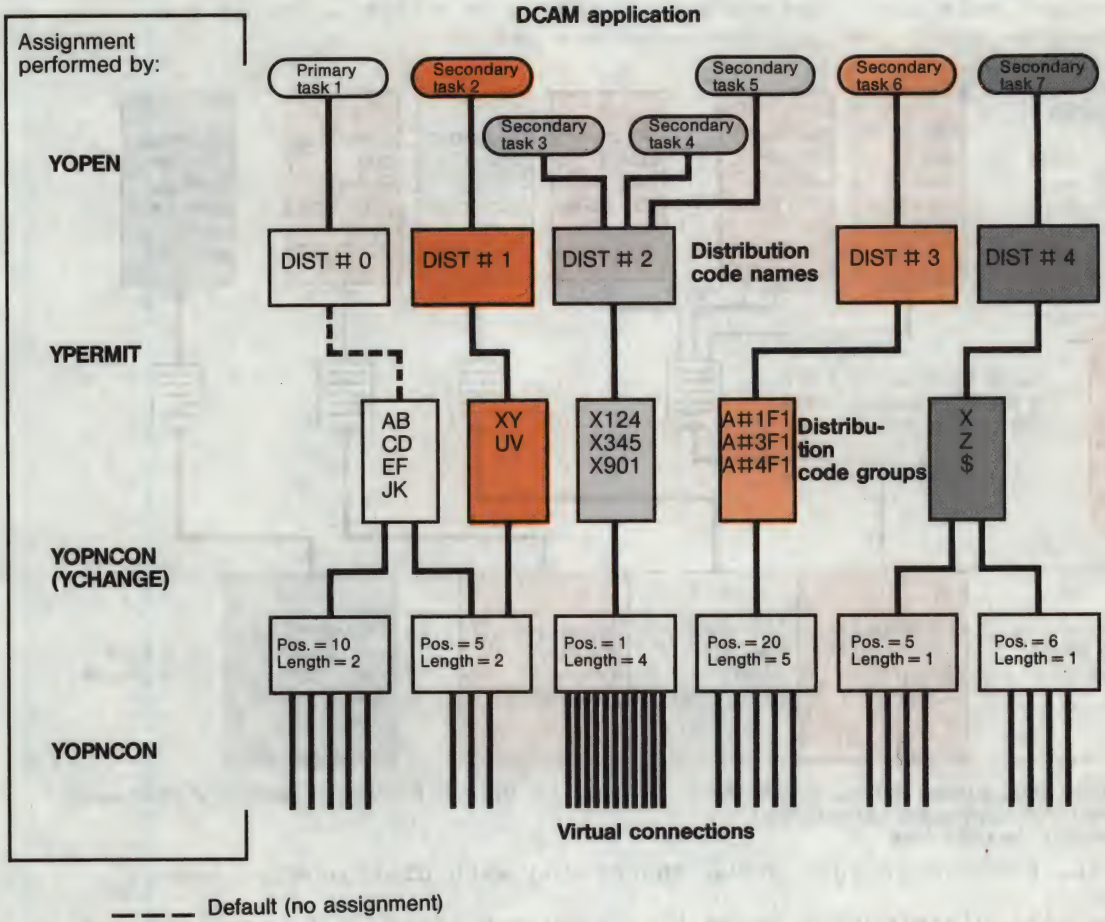


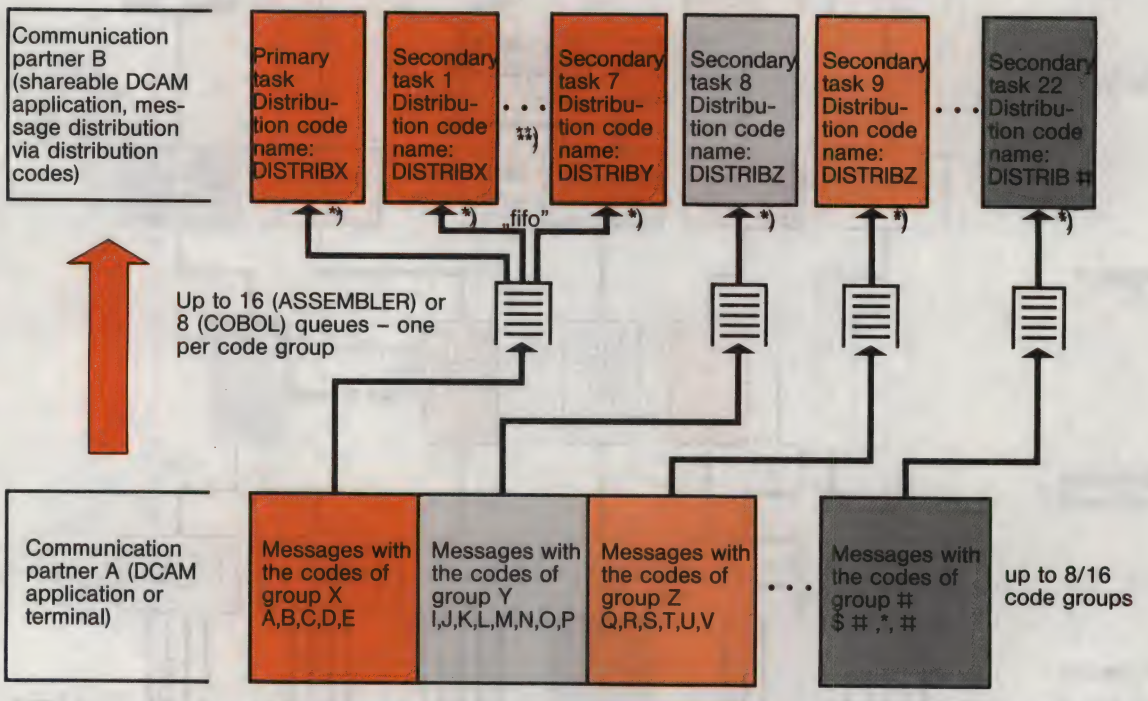
Fig. 3-17 Summary of message distribution with distribution codes (example)

3.3.5.1 Assignment of a Distribution Code Name to Distribution Code Groups

The YPERMIT call assigns a distribution code name to a distribution code group. Once this assignment has been made, tasks that have defined this name can receive messages with the codes of the assigned group.

The following applies:

- A Group of distribution codes (stage 2 definition) can be assigned to only **one distribution code name.**
- Up to **8 tasks** can use **the same distribution code name.** This becomes necessary when a single program controls several tasks (SHARED CODE, see 4.1.6 and 4.2.4). Tasks with identical distribution code names are served in accordance with the FIFO principle (FIFO=first in - first out: the first entry in the queue will be the first one processed; see also Fig. 3-18).
- Depending on the assignment made, a task can also access the messages in an **originator-oriented** way (YRECEIVE SPEC). It is not necessary to select this queue beforehand (there is no CS/CA state of the connection when distribution codes are used).



*) The queues (code groups) are assigned to the distribution code names with separate data communication function calls.

**) Up to 8 tasks per distribution code name

Fig. 3-18 Example of task group addressing with distribution codes

Messages with distribution codes that were not assigned to a distribution code name or that cannot be interpreted are directed to the primary task.

Old assignments can be replaced by new ones when a new YPERMIT call is issued.

3.3.5.2 Cancellation of the Assignment

If a new assignment is not to be made for a distribution code name but the existing assignment is to be canceled, the YFORBID call can be used. Data which has already been received is then transferred to the primary task.

3.4 NAME ASSIGNMENT FUNCTION

This function permits parameters for the DCAM application or the virtual connection to be specified at run-time only. This is achieved by linking the current values with those specified in the program in a task-oriented table (CLT=communication link table). (By comparison, the entries in the FCB (file control block) of a program are updated using the TFT (task file table) whose entries were generated by the FILE macro or the /FILE command.)

Fig. 3-19 shows an example (for ACB control block see 4.1.1; for A structure see 4.2.1).

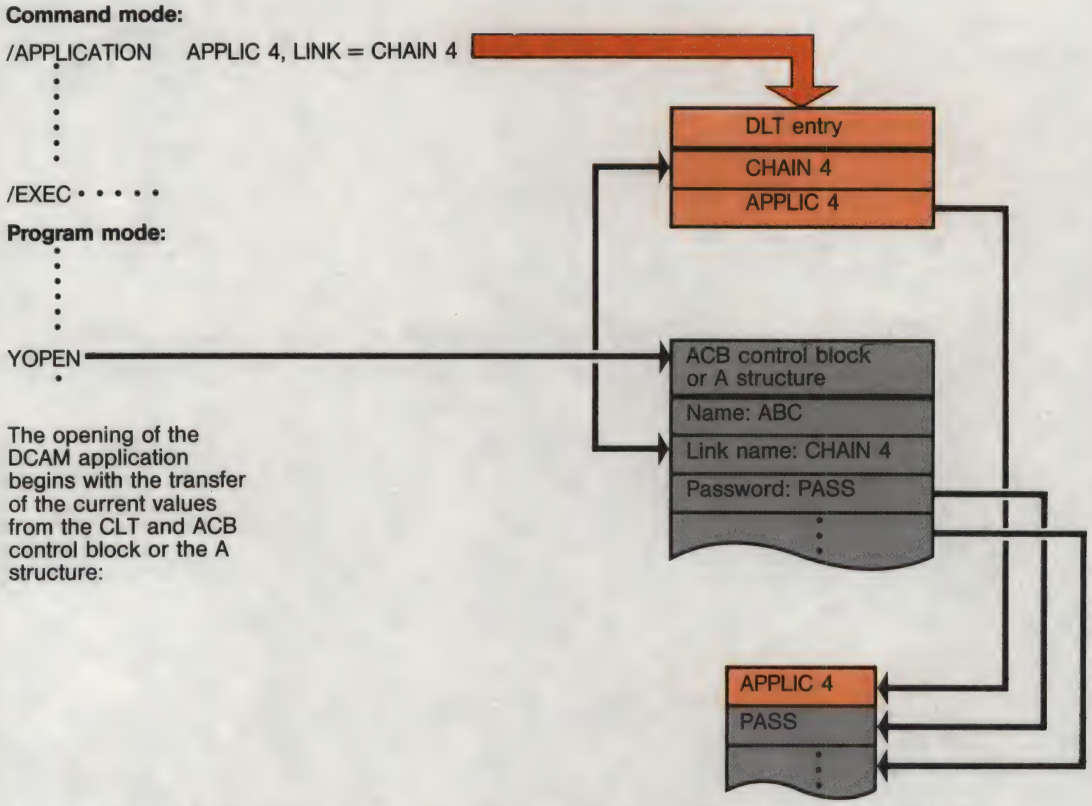


Fig. 3-19 Example of assignment of a DCAM application name

The table is set up:

- in the progam mode (ASSEMBLER only) by means of the

  YAPPL macro for entries on the application, and the

  YCONN macro for entries on the connection

- in the command mode (ASSEMBLER/COBOL) by means of the

  /APPLICATION command for entries on the application and the

  /CONNECTION command for entries on the connection (see "Control System Command Language").

The linkage is established by specifying a **Link name** both in the program and in the CLT. The following values can be updated in this fashion:

- **For a DCAM application:**
  - the name of the DCAM application;
  - the distribution code name;
  - the password for the connection of a secondary task to an application, as specified in the primary task
  - the password for the connection of a secondary task to an application, as specified in the secondary task;
  - the password for establishing a connection, as defined in the primary task;

- **For the virtual connection:**
  - the name of the partner;
  - the name of the processor node to which the partner is connected as a station;
  - the accompanying information;
  - the password for connection establishment, as specified by the requesting task.

When the YOPEN or YOPNCON call is executed, the specifications are entered in DCAM control blocks or data structures. They are updated with the values in the CLT providing it contains entries. If it does not, the original values remain unchanged. Following this, dynamic name assignment for this application or virtual connection is possible again only after it has been closed (except for alterations of the accompanying information by means of YCHANGE (see 3.2.4).

If CLT entries are to be deleted, the YAPPL or YCONN macro or the corresponding commands with just the link name should be issued.

4 CODING AND EXECUTION OF DCAM PROGRAMS

This chapter does not contain exact coding instructions since separate manuals are provided for this purpose (see "DCAM Macros" and "DCAM COBOL Calls").

Instead it describes the specific language-related characteristics of the DCAM interface. The DCAM programmer can use two languages:

- ASSEMBLER
- COBOL

After the use of DCAM in these languages has been outlined, the following questions will be answered:

- How is a DCAM program started?
- How is a DCAM program terminated?

4.1 WRITING AN ASSEMBLER PROGRAM

The ASSEMBLER program interface provides the full range of DCAM functions as described in detail in chapter 3.

The purpose of this section is to provide an overview of the specific interface characteristics.

4.1.1 Macro Calls and Control Blocks

All DCAM functions are activated by macro calls. The action macros always refer to control blocks, in which all the parameters are stored. A distinction is made between two groups of calls depending on the type of control block they refer to:

- macro calls that refer to an **ACB (application control block)**, and
- macro calls that refer to an **RPB (request parameter block)**.

The following calls refer to an **ACB**, which contains all the information on the DCAM application:

- **YOPEN** (Open a DCAM application)
- **YCLOSE** (Close a DCAM application).

If asynchronous DCAM notifications are to be processed, an ENB **(event notification block)** is required in addition. The identifiers for the contingency routine are stored in this block (see 4.1.5).

Fig. 4-1 provides an overview.

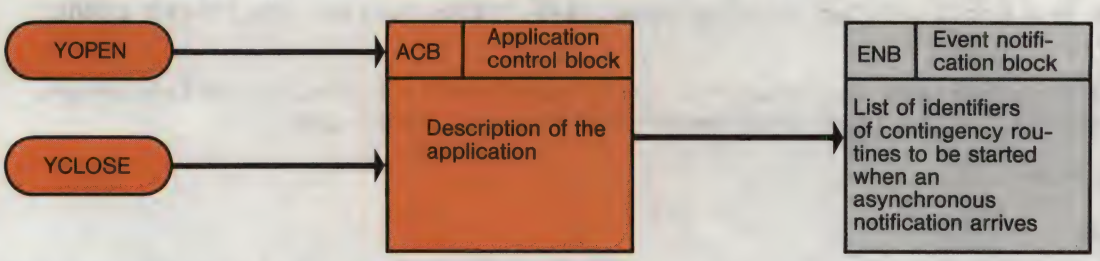


Fig. 4-1 ACB-related macro calls

The following calls refer to an **RPB** containing the current parameter values of the action calls:

- **YSETLOG** (Change the state of a DCAM application)
- **YINQUIRE** (Inquire for entries about DCAM applications, partners and connections)
- **YOPNCON** (Open a connection)
- **YCLSCON** (Close a connection)
- **YREJLOG** (Reject a logon request)
- **YCHANGE** (Change the characteristics of a connection)
- **YSEND** (Transmit a message)
- **YRECEIVE** (Receive a message)
- **YSENDREC** (Transmit a message and receive a new message)
- **YRESET** (Delete receive messages and change the connection state CS/CA)
- **YPERMIT** (Allow a message to be received with a distribution code)
- **YFORBID** (Prevent a message being received with a distribution code).

Further control blocks are required in addition to the RPB for various calls:

The parameters for a connection are located in the **CCB (connection control block),**

the position and length of the distribution code, the code sign and the references to assigned distribution code groups are located in the **DIP (distribution parameter block),**

and the distribution codes themselves are described and the **DCG (distribution code group block).**
Fig. 4-2 provides an overview.

Note:

DCAM users connected to a packet switching network (e.g. Datex-P) via an an X.25 interface must note that certain restrictions apply to calls relating to connections or data transmission. They are described in the manual "DCAM Macros" (Appendix A.11).

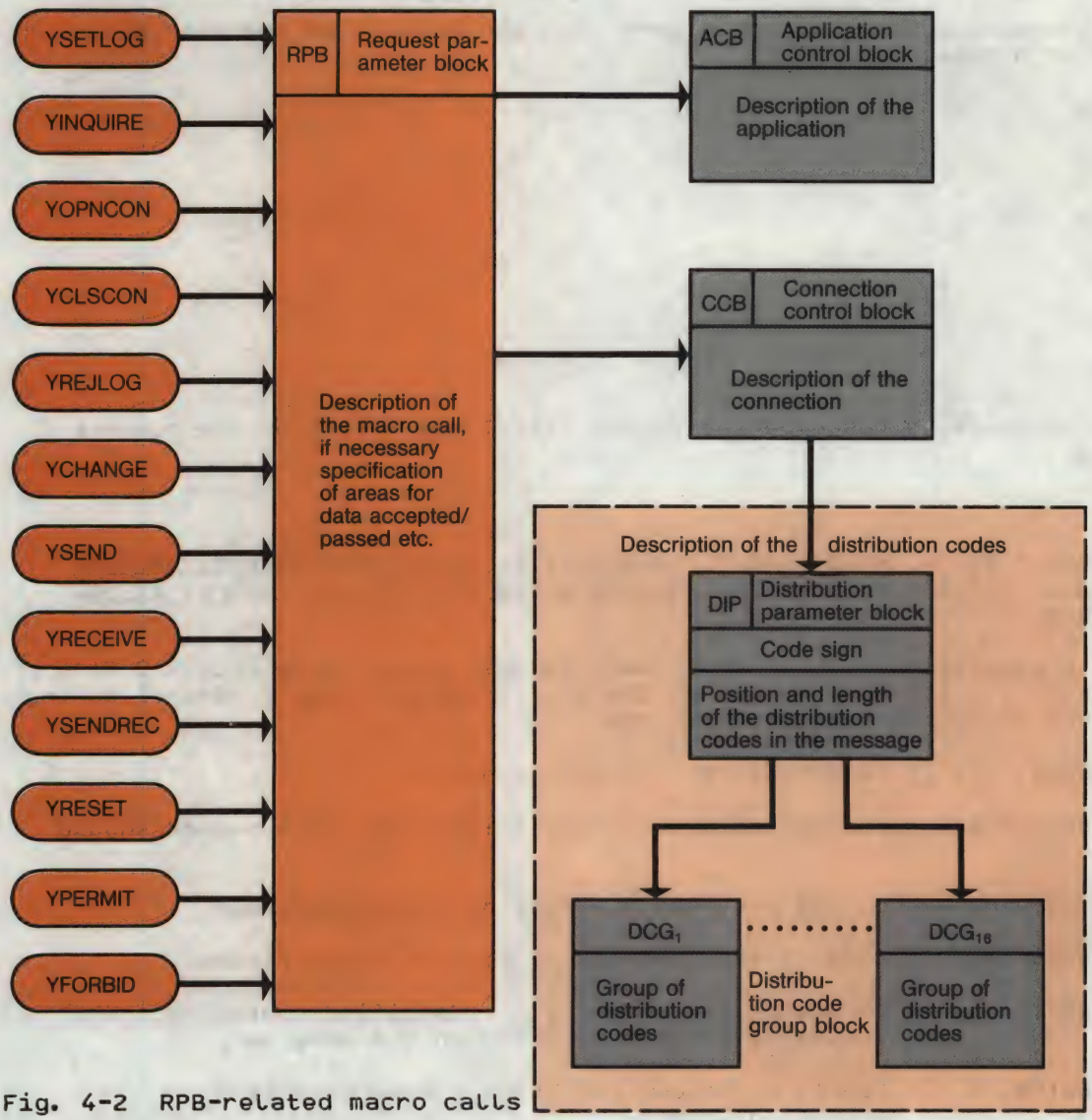


Fig. 4-2 RPB-related macro calls

The programmer need not know the internal structure of the control blocks as macro calls are available to him for the generation and handling of control blocks.

The following macro calls are provided for **static control block generation:**

- **YACB** (Generate ACB)
- **YCCB** (Generate CCB)
- **YENB** (Generate ENB)
- **YDIP** (Generate DIP)
- **YRPB** (Generate RPB)
- **YDCG** (Generate DCG)

In this case the control blocks are generated during assembly.

Note:

Later modifications of the block structure by the manufacturer may necessitate reassembly.

The control blocks can also be generated and handled by the user directly with macro calls

YDDACB

YDDCCB

YDDDCG

YDDDIP

YDDENB

YDDRPB

These macro calls describe the layout (DSECT and CSECT) of the control blocks.

Note:

With these macro calls source compatibility is not guaranteed, because the layout of the control blocks is changed if this is required by new versions.

Dynamic generation of a control block in user memory area (class 6 memory) or optionally in an area outside the user program (class 5 memory) during the program run is possible with the

- **YGENCB** (Generate control block) macro.

The program being unaffected by later modifications, this method should be used.

The following macros are provided for **control block handling:**

- **YMODCB** (Modify the contents of control block fields),
- **YSHOWCB** (Transfer the contents of identified control block fields to an area reserved in the program),
- **YTESTCB** (Compare the control block field contents with specified values).

The RPB can also be modified with any macro call that addresses it. It is up to the user either to create several RPB control blocks or always to address the same one and update its contents.

The application characteristics described in the ACB are transferred to DCAM with the YOPEN call. After YOPEN execution, DCAM returns an **AID (application identifier)** in the ACB. From then on all calls referring to this application may use the address of the ACB control block containing this identifier (see Fig. 4-3).

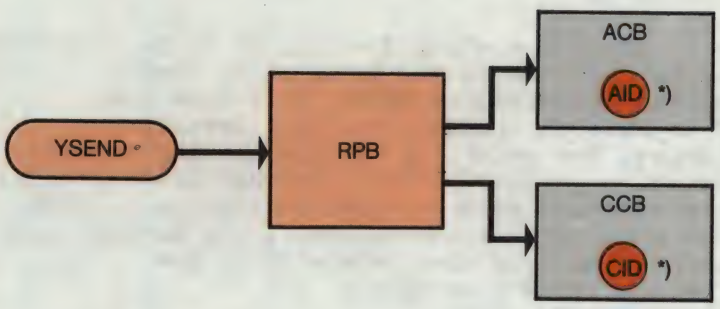


Fig. 4-3 Example of control block addressing without identifiers

It is also possible to save the AID and to enter it in the RPB used for further calls (Fig. 4-4).

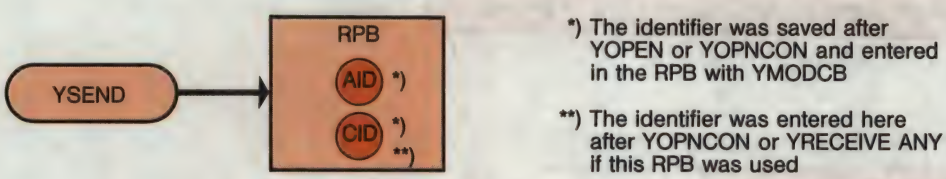


Fig. 4-4 Example of the addressing of an RPB in which valid identifiers were entered.

In this case the memory area for the ACB can be used for other purposes. The same applies to the description of the connection in the CCB to the DIP and the DCG. The **CID (connection identifier)** returned in the CCB and RPB can be used after an YOPNCON, for example. YSEND and YRECEIVE or YSENDREC then only require RPB control blocks containing the relevant AID and CID identifiers. If the entry was not made by DCAM, it can be made with YMODCB provided the identifier was previously saved with YSHOWCB.

The use of the identifiers is advantageous because DCAM finds all the information required for the call in the RPB. This speeds up processing.

4.1.2 Synchronous Execution of DCAM Calls

If a call is sent to DCAM with an operand stating that it is to be executed synchronously, control can only be returned to the program by task management after this call has been processed by DCAM. This does not present any problems when DCAM is able to process the call immediately. However, with some functions, **delays** may occur due to DCAM having to wait for a response from the communication partner. When a virtual connection is being set up (YOPNCON), the communication partner must contribute to this operation and when messages are to be received (YRECEIVE), the partner must have sent them and they must be available in the data area of the communication system and have been entered in a queue.

The programmer can decide whether and how long he is prepared to wait until the call can be executed. He defines the maximum waiting time for each call. He can also specify that he is not willing to wait. In this case the call is terminated immediately even if the instruction cannot be performed. The call must then be repeated later in the program if required (cf. Fig. 4-5).

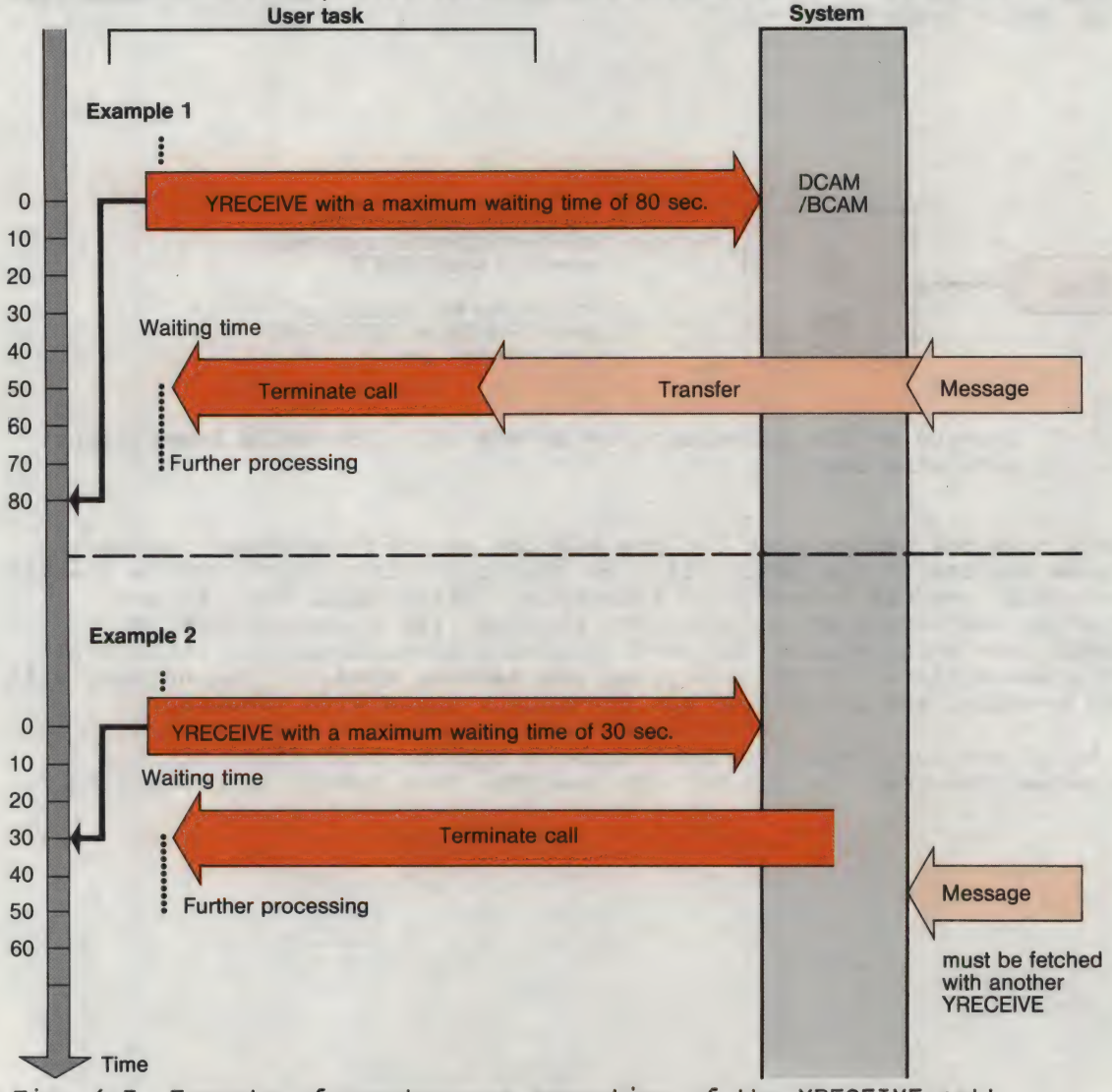
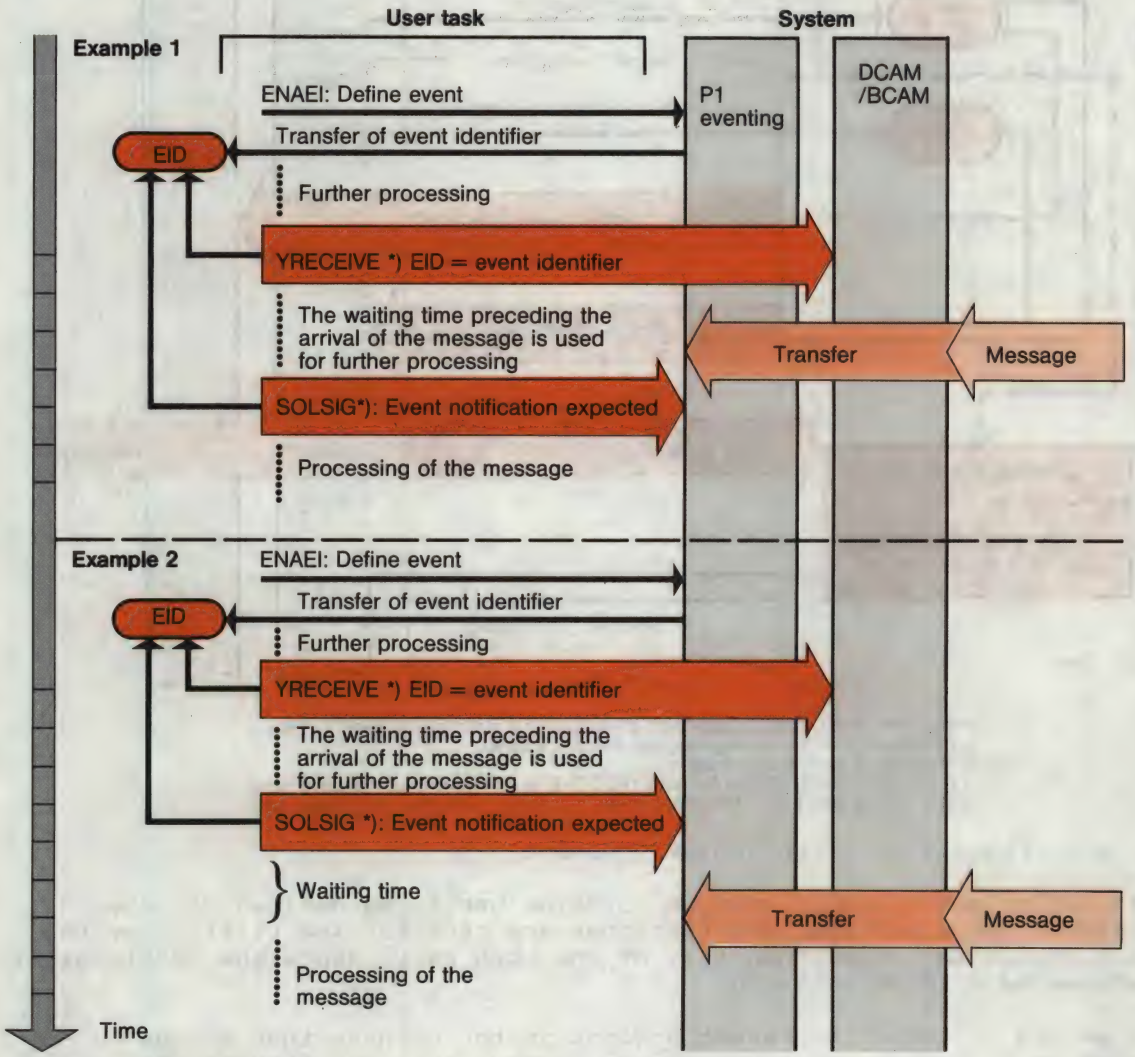


Fig. 4-5 Example of synchronous execution of the YRECEIVE call

4.1.3 Asynchronous Execution of DCAM Calls

As mentioned in the preceding section, connection establishment and message reception may lead to delays. In order to increase data throughput, particularly if a very large number of partners are to be serviced, these delays can be used for other processing. For this purpose, the user can define an event (ENAEI macro) and specify its identifier EID in the YOPNCON, YRECEIVE or YSENDREC macro whose instruction is to be executed asynchronously. Control is then returned to him immediately after macro termination and instruction acceptance by DCAM. The event may be, for example, message transfer. The user can test for this event at that location in the program (SOLSIG macro) at which he wishes to process the message, for instance (see Fig. 4-6).



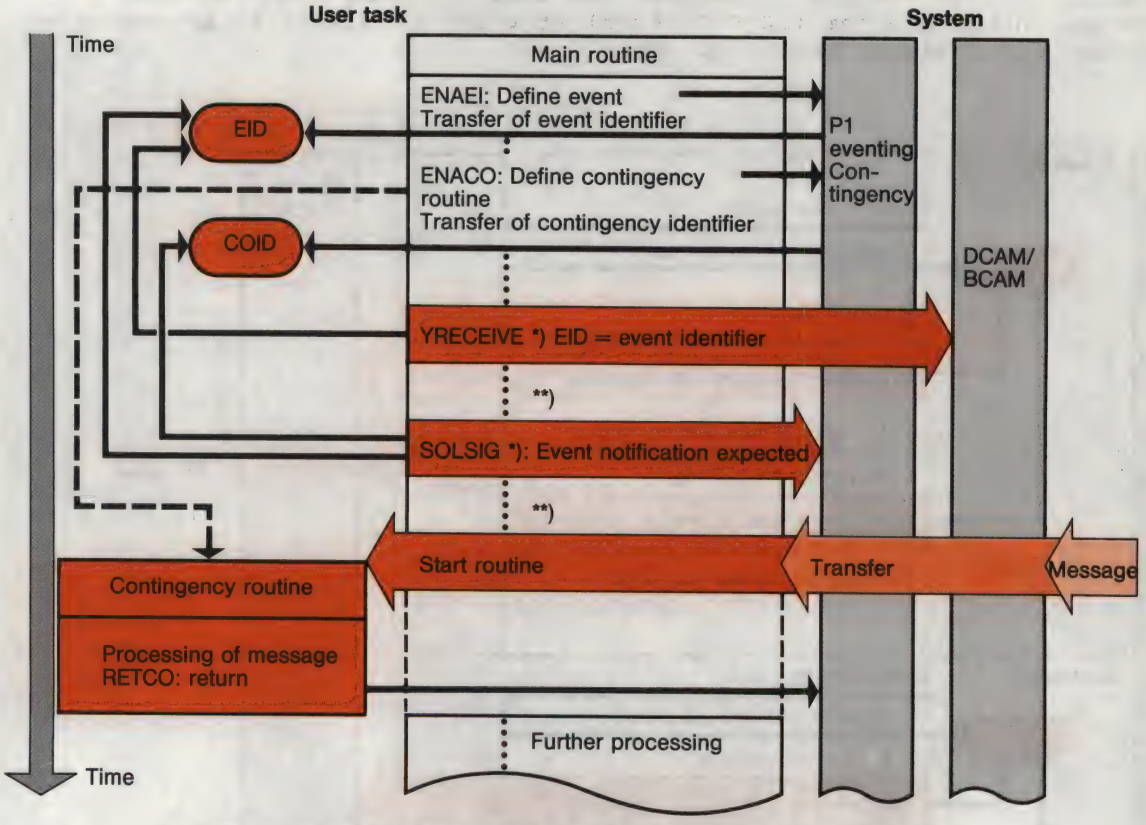
EID = Identifier of the event (in this case, the arrival of the message), is used as the reference point for message transfer and acceptance
*) A maximum life is defined for YRECEIVE and SOLSIG in each case (not indicated here).

Fig. 4-6 Example of asynchronous execution of the YRECEIVE call

At this point he can wait, if necessary, for a certain period of time, if for instance, the message has not yet arrived.

He can, however, also link the event with a contingency routine. In this case, the user should test for the event as soon as he has issued the DCAM call. He can do this any time provided he has defined the contingency routine and has received its identifier COID (ENACO macro).

When the event occurs, the routine is started automatically (see Fig. 4-7).



EID = identifier of the event (in this case, the arrival of a message), is used as the reference point for message transfer and acceptance.
COID = identifier of contingency routine
*) A maximum life is defined for both YRECEIVE and SOLSIG (not indicated here)
**) The waiting time before the arrival of the message is used for other processing

Fig. 4-7 Example with contingency routine

It should be noted that a maximum lifetime has to be defined for the individual calls and that the lifetimes are related. The lifetime of the SOLSIG should be longer than that of the DCAM call, since the SOLSIG might otherwise have to be repeated.

Care should be taken to prevent actions in the program that assume completion of asynchronous processing without this actually having taken place (closing of the connection, for instance, while a YRECEIVE is being processed asynchronously).

Restriction:

Up to 8 instructions of the same type can be processed asynchronously at a given time (see Appendix A.2).

In order to be able to access values which were valid when the call was issued or to transfer the address of the control block used, the user can define event information.

This takes place when YOPNCON, YRECEIVE or YSENDREC is issued. It is transferred when the event has arrived, either in a defined field (no contingency routine) or in a register (when a contingency routine has been defined).

4.1.4 Feedback Information

The user receives feedback information after the termination of a DCAM call. This information consists of a 4-byte code which is entered in a field of the ACB or RPB control block, the so-called FDBK field, and in register 15 (see Table 4-1).

FDBK/Register 15			
Byte1 (leftmost)	Byte2	Byte3	Byte4
FDB1: Feedback summary	FDB2: Reason for rejection (error code)	FDB3: Condition codes	FDB4: Indicators

Table 4-1: Feedback information format

The feedback information is made up of several bytes. By evaluating byte 1 (FDB1) the user will have sufficient information to estimate roughly what needs to be done. The user can either evaluate byte 2 (FDB2) or make do with a listing that permits later evaluation. The job on hand will ultimately be the decisive factor in this matter.

Bytes 3 and 4 (FDB 3-4) contain additional information, e.g. message format, transport acknowledgment or message, etc.

The feedback information is always in the FDBK fields of the appropriate control blocks.

In the case of a **synchronous call**, the feedback information includes information on the processing or rejection of the call.

In the case of **asynchronously** executed **calls**, the feedback information only includes information on instruction acceptance or rejection. Register 15 and the FDBK field cannot yet contain any values relating to the execution of the instruction. The RPB control block will contain further information only after the instruction is executed. In this case, the user can locate the RPB by transferring the address of the RPB in the EIDREF field. After a SOLSIG call the user receives the RPB address via the RPOSTAD field or register 3 (contingency). Then he can access the RPB and with YSHOWCB he can access the feedback information in the FDBK field.

The feedback information from the SOLSIG call (register 15) or the contingency routine (register 2) informs the user whether the event which occurred was the expected one or whether an error or a timeout occurred, but information on the execution of the call by DCAM is not provided (see SOLSIG in the manual "BS2000 Executive Macros").

A number of calls (YOPNCON, see 3.2.1.2 and 3.2.1.3; YRECEIVE, see 3.2.2) provide further information besides the feedback information (see the corresponding sections (chapter 3) on the functions) but only if the feedback indicates no errors (i.e. FDB1=X'00').

4.1.5 Asynchronous DCAM Notifications

Asynchronous notifications can be issued to the task by DCAM for a number of events in the data communication system which may occur in asynchronous relationship to program processing and which may have a decisive effect on processing (see Table 4-2).

Notification	Transmitted by DCAM	Related to
LOGON	only to the primary task	connection function
PROCON		
SECOND		existence function
LOSCON	to primary or secondary task	connection function
COMEND		existence function
EXPR		data transmission function
TACK		

Table 4-2: Asynchronous DCAM notification

These **events** are:

LOGON:

- A connection request has arrived.

LOSCON:

- A connection was closed by the communication partner, the operator or due to an error.
- A connection is about to be closed (warning); (see /BCON and /BCTIMES commands in "Network Management in BS2000").

Note:

If the LOSCON event occurs without a warning, the connection is already cleared down; if the user now enters the macro call YCLSCON, it will be rejected with a return code ≠ 0.

PROCON:

- Partners already defined at communication access system generation time (XSTAT macro in "Generation of a Data Communication System") were proposed for logon in the YOPEN or a /BCIN operator command.

SECOND:

- A secondary task was opened or closed.
- The distribution code of a message is assigned a distribution name which is not opened by any secondary task. The primary task must first issue YFORBID for the distribution name before it can receive the message proper.

COMEND:

- The communication access system was terminated.
- Termination is pending (warning) (see /BCTIMES command in "Network Management in BS2000").
- The DCAM application will be terminated shortly (warning for connected secondary task) (see /BCTIMES command).

EXPR:

- An express message has arrived.

TACK:

- A transport acknowledgment has arrived.

The DCAM application can be notified of the event concerning it if this is specified when the application is generated/opened. As the event notification results in a contingency routine being executed, a **contingency routine** must be **defined** for a notification to be received. The **ENACO macro** can be used for this purpose (see "Executive Macros"). The returned identifier (COID) has to be transferred to DCAM via the ENB event control block (see 3.2.1). The notifications which are to be accepted and the contingency routine which is to be initiated have to be defined at YOPEN time for the duration of an application.

Primary tasks are notified of all events, secondary tasks only of LOSCON, COMEND, EXPR and TACK. If a contingency routine is initiated, registers 1 to 7 contain all the information required for the processing of the event. The other registers have no defined contents. Input to the base register contents is the user's responsibility. Access to register contents of the interrupted routine or of the main routine is possible with the CONTXT macro (see "Executive Macros"). The priorities of the contingency routines are established when they are defined (ENACO) and can be modified with the LEVCO macro (see "Executive Macros").

DCAM expects a response to each notification. If no response occurs, this is interpreted as a rejection of the notification. The response does not have to occur within the contingency routine itself, just within a certain period of time defined during the generation of the communication system (see /BCTIMES command and Fig. 4-8).

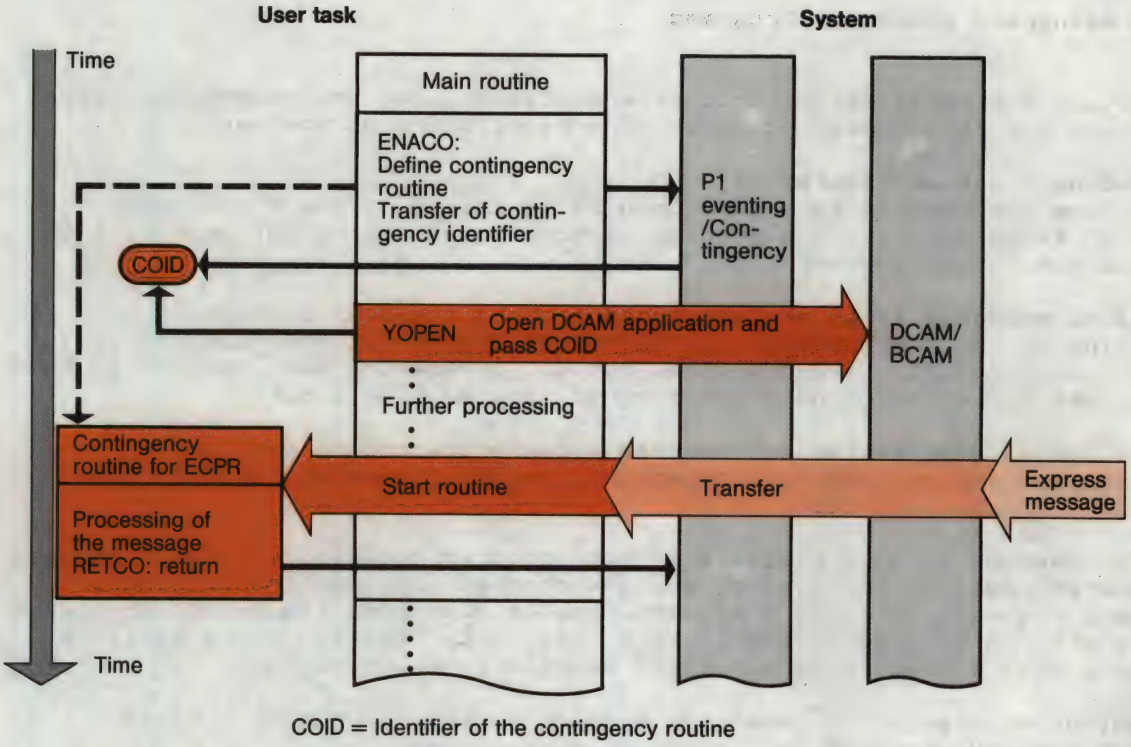


Fig. 4-8 Example of express messages being received via an asynchronous DCAM notification (EXPR)

The decision during the generation/opening of the DCAM application **not to accept** certain **notifications** has the following consequences:

- **LOGON** is not defined: Connection requests issued by the communication partners can only be processed by means of a YOPNCON call issued during the program run.
- **LOSCON** is not defined: Notification of connection loss can be obtained at the earliest from the feedback information from a call which refers to this connection.
- **SECOND** is not defined: The primary task must use other means (PI eventing) to discover that a secondary task has opened the application and which distribution name it has.

  The primary task is unaware of messages assigned a distribution name without a connected secondary task. These messages are not received by any task and are deleted after expiry of a system monitoring period.
- **PROCON** is not defined: Pre-defined logon proposals are not reported to the DCAM application.
- **COMEND** is not defined: The fact that the DCAM application or the communication access method no longer exists is indicated in the feedback information at the earliest when the next call is issued to DCAM.
- **EXPR** is not defined: Express messages must be fetched with YRECEIVE in the order of the incoming messages.
- **TACK** is not defined: Transport acknowledgments must be fetched with YRECEIVE in the order of the incoming messages.

4.1.6 Reentrant ASSEMBLER Programs

In BS2000, a program can control several tasks, i.e. the program is loaded only once for this number of tasks. The results are as follows:

- **Loading times are shortened**
  because the program is loaded when it is called by the first task. When other tasks call it, the program is not loaded again, but used by them also i.e. it is shared.

- **Dialog response times are shortened** because the DCAM application, acting as the communication partner, has shared out the "burden" of incoming inquiries among several tasks. Timesharing task control is thus implemented although only one program is used (see 2.3.1).

- The **system utilization is improved** because the paging rate is lower and the program is managed only once for several tasks in the main and page memories.

This processing is only possible if such program modules, which are managed as **'shared code'** in the system, are invariable, i.e. reentrant. Since, however, task-specific I/O areas etc. are usually also required, this means that a program must be divided into a 'read-only' section and a variable section. This procedure is supported by DCAM as follows:

- **Control blocks** can be generated during the program run in a task-oriented area managed by DCAM or in a user area.

- By means of the **MF parameter** the parameter list and the operation code (SVC) of the macro can be separated for calls generating and handling control blocks.

- **Register notation** is used to a large extent.

Table 4-3 and Fig. 4-9 show the possibilities available through reentrant programming and the required subdivision of the program.

Program section	Invariable (shared code)	Variable (task-oriented)
Instructions	- Invariable instruction code (using EX instruction, if necessary) - Addressing of the variable section via registers (MF=E) - If required, provision of memory for the variable section (Fig. 4-9, model B)	- Instruction code, including input to registers for branch to invariable module (Fig. 4-9, model A)
Data and areas	- Number constants - Text constants - Address definitions (DSECT) - Parameter lists (MF=L)	- I/O areas - Computation fields - Save areas (registers) - Parameter lists (MF=L)

Table 4-3: Shared code program structure

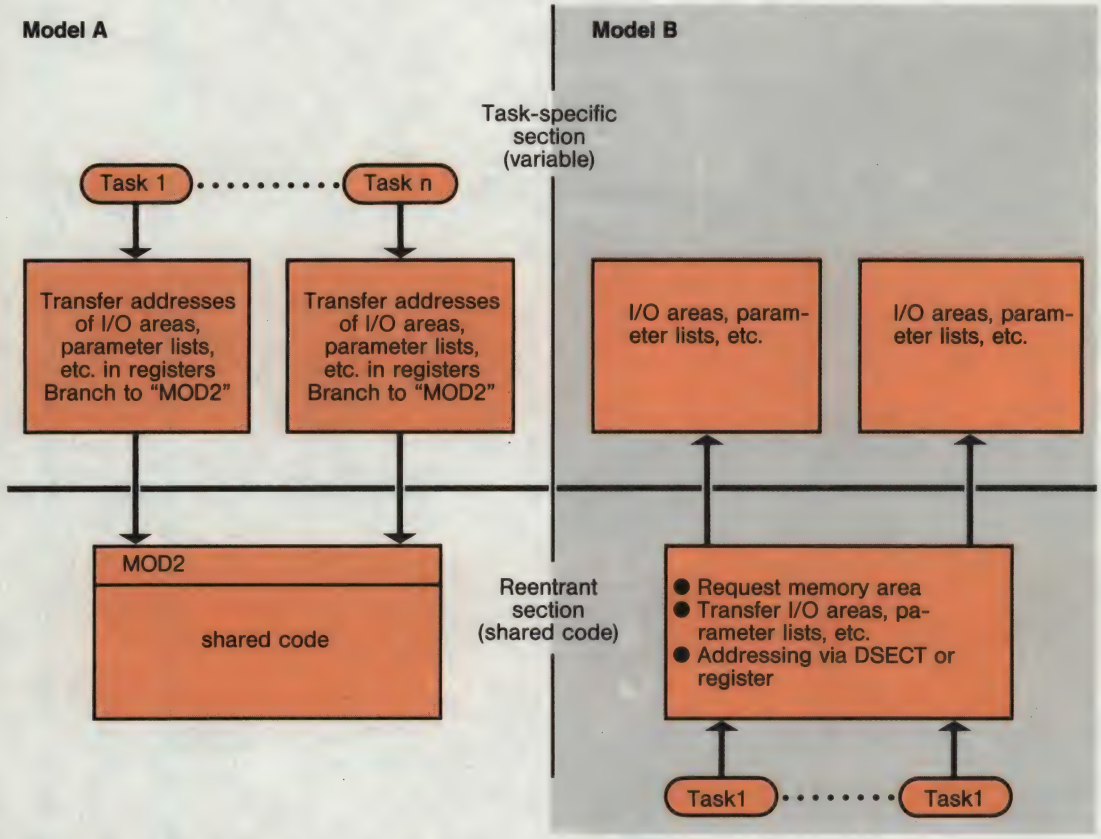


Fig. 4-9 Models for shared code programming

4.1.7 Use of Other System Interfaces

There are **no limitations** on a DCAM ASSEMBLER program using other system interfaces. Some interfaces or interface characteristics are in fact particularly suited to solving specific transaction mode problems.

For the purpose of data transfer between tasks and the synchronization of tasks the user is provided within the framework of the Executive macros with a "common memory" with "task serialization" and "inter-task communication". Event-driven processing is possible when using the P1 eventing and contingency interface.

File processing with the access methods of the data management system DVS is possible in the SHARED UPDATE with ISAM and USER PAM; i.e. access of a task group to a file is supported with the necessary protection mechanisms.

4.2 GENERATION OF A COBOL PROGRAM

A series of access modules to DCAM functions was provided for use in COBOL programs, which permit synchronous processing. COBOL programs can control primary tasks in simple user problems, but are also suited to controlling secondary tasks especially in more complicated problems (see also 2.3).

4.2.1 COBOL Calls and Data Structures

All calls to DCAM are formulated as branches to subroutines (CALL...); the associated data structures with the parameters and data fields must be declared in the WORKING-STORAGE SECTION.

Six different data structures are provided for:

- **Application structure (A-Struktur):**

  Description of the DCAM application; must be present at least once in the program.

- **Connection structure (V-Struktur):**
  Description of the virtual connection; either for each connection or just once for several connections.

- **Instruction structure (B-Struktur):**

  Description of an instruction; either for each instruction separately or just once for several instructions.

- **Distribution structure (VTLG-Struktur):**

  Description of message distribution with distribution codes; only necessary if message reception is to be controlled in this way.

- **Wait structure (W-Struktur):**

  Description of the operands causing the system to wait for the termination of asynchronous CALLs.

- **Data structure (FHS-Struktur):**

  This is only necessary if the data are to be formatted with the integrated FHS C interface. (This requires the software product FHS from Version 3.0 onwards). The FHS modules must be available on the user TASKLIB.

COPY elements are available for all data structures:

YDDCUAPL	for the A structure
YDDCUCON	for the V structure
YDDCUCOM	for the B structure
YDDCUDIS	for the VTLG structure
YDDCUWAI	for the wait structure
FHSMAINP	for the FHS parameters

Table 4-4 shows which data structures are used in the calls of the individual subroutines. For some calls an additional area for data acceptance and transfer is required.

Some calls require still further areas which are not shown here (refer to "DCAM COBOL Calls").

Function	Subroutine	Data structures					Area	FHS parameter
		A	V	B	VTLG	W		
Existence function	YOPEN	x						
	YCLOSE	x						
	YSETLOG *)			x				
Connection function	YOPNCON *)	x	x	x	[x]			
	YCLSCON *)	x	x					
	YCHANGE *)	x	x		[x]			
	YREJLOG	x	x					
Data communication function	YSEND	x	x	x			x	[x]
	YRECEIVE	x	x	x			x	[x]
	YPERMIT *)	x			[x]		x	
	YFORBID *)	x					x	
	YRESET	x	x	x				
Wait function	YWAIT	x	x	x		x		

*) only callable with primary task

Table 4-4: Subroutines and data structures

The subroutines perform the following functions:

YOPEN: Opens a DCAM application

YCLOSE: Closes a DCAM application

YSETLOG: Changes the state of a DCAM application

YOPNCON: Sets up a virtual connection

YCLSCON: Closes a virtual connection

YCHANGE: Changes the characteristics of a connection

YSEND: Transmits a message

YRECEIVE: Receives a message or transport acknowledgment

YPERMIT: Allows data to be received via distribution code

YFORBID: Prevents data being received via distribution code

YREJLOG: Rejects a connection setup request

YRESET: Resets pending YRECEIVE calls

YWAIT: Waits for some DCAM event

A special convention applies to **status check** when this concerns the existence or connection function. **YINQUIRE** is used to call for system testing. The following inquiries can be made:

- **Which partner** wishes to log on next (with connection message possibly received in LGMSG)? (TOP)
- **How many partners** wish to log on or have logged on? (CNT)
- **What is the state** of a certain DCAM application? (APP)
- **What characteristics** does a partner have? (PTN)

The A and B structures are used, and an area must be specified. The layout of this area varies depending on the function of the call. The function is defined in a special field (Table 4-5).

Function	Subroutine	Function of YINQUIRE	Data structures A	Data structures V	Areas *)	LGMSG
Status inquiry for the existence and the connection function	YINQUIRE	TOP CNT APP PTN	X X X X	X X X X	X X X X	[X]

*) Area depends on function

Table 4-5: Status testing

Note:

The subroutines may also be called using the first four letters in a name, e.g. YOPN instead of YOPNCON.

To simplify programming, it is recommended to define the data structures once in a **source code file.** The COBLUR utility routine can be used for the creation of a COBOL source program library. By means of the COPY statement, the stored data structures can be transferred to the program and, if necessary, modified. Information on program compilation is provided in a separate manual.

4.2.2 Execution of the Calls

4.2.2.1 Synchronous Execution

In synchronous execution the next instruction after the branch to the subroutine is only executed once the DCAM call has been processed. For calls in which delays are likely, e.g. YOPNCON: the wait for a connection setup request or acceptance of the connection request by the partner, or YRECEIVE: arrival of the message from the data communication system, a maximum wait period can be defined. The call is terminated after the specified period. If there is to be no wait, a "tentative" call is issued which must be repeated if necessary.

4.2.2.2 Asynchronous Execution

Connection setup and the reception of messages may lead to delays (see 4.2.2.1). In order to increase data throughput, particularly when a large number of partners need to be serviced, the delays arising may be utilized for additional processing. The YWAIT enables the user to wait for some event. Once this event has occurred, he can resume processing.

Possible events:

- OPENED — the YOPNCON request has terminated
- LETTER — the YRECEIVE job has terminated
- GOSIGNAL — the memory bottleneck has cleared
- LOSCON — the connection was cleared by the partner or the system
- NOEVENT — no DCAM event occurred during the wait period

4.2.3 Feedback Information

After the return from a subroutine, i.e. after execution of a DCAM call, feedback information is stored in 3 fields of the data structure used (application structure or instruction structure).

The feedback information is subdivided into 3 fields:

- **return code**
- **error code**
- **indicator**

The return code provides a summary of the information given in more detail in the error code and the indicator. It will be needed, for instance, in order to branch to an error routine.

The error code must then be saved in this routine so that it can be evaluated.

Following the execution of YRECEIVE; the indicator contains additional information on the type of transport acknowledgment (positive or negative) or the message class (element, subgroup, group; normal or express message).

Furthermore, entries are made in 3 fields of the instruction structure following **YRECEIVE**:

- the **sequence number** of the message received
- the **sequence number of the acknowledged message** (if a transport acknowledgment was received)
- the **actual length** of the message received (even if the input area was smaller)

Additional information is passed in the event of extra calls (YOPEN, YOPNCON, YWAIT) - provided the acknowledgment does not indicate any errors (see the appropriate sections (chapter 3) of the functional description).

4.2.4 Reentrant COBOL Programs

The essential condition for the reentrant quality of a code is that the code was not modified in the course of execution (cf. 4.1.6). Only then can it be managed and used in the system as shared code.

In employing overlay techniques, the COBOL compiler creates a root segment that is reentrant and various overlays that are also, subject to a few restrictions, reentrant. The root segment contains the V constants for linking both the overlays as well as the COBOL runtime system (ITL...) and the CALL modules. This segment is assigned the name given in the PROGRAM-ID.

The overlays are called by the root segment via PERFORM. Their names are made up of the first three characters of the name given in the PROGRAM-ID, one special character (#) and the overlay number (for instance, ABC#50).

In these overlays, no module may be invoked with a CALL that is not reentrant. This is, however, true of DCAM COBOL modules. Thus DCAM calls may not be issued in such overlays.

In addition, the following should be noted:

When using the COB1 compiler, reentrant code is generated in overlay segments.

When using the ANSICOB compiler, the following restrictions hold:

- no accessing of data items defined with OCCURS ... DEPENDING ON ...
- no GO TO together with ALTER allowed
- no DISPLAY possible
- no ACCEPT possible.
  Otherwise, reentrant code is generated.

Fig. 4-10 illustrates segmentation in a sample COBOL program.

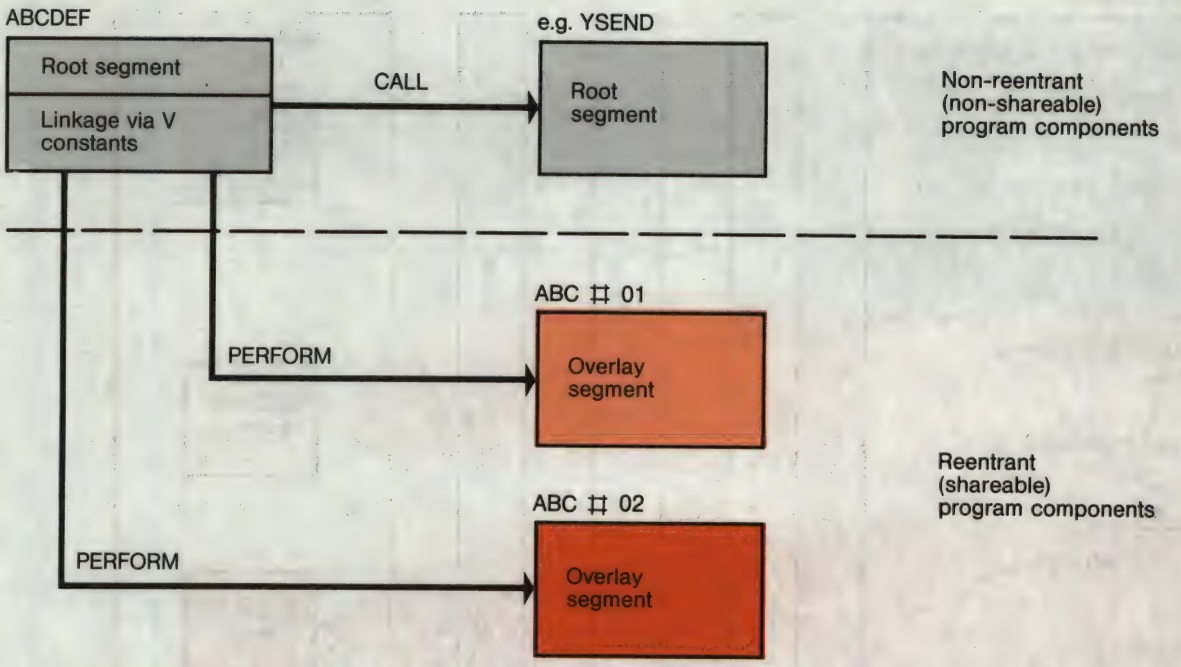


Fig. 4-10 Segmentation of a COBOL program

In order to test overlay segments for reentrancy (read-only attribute), the TRAITS statement must be employed when they are entered in a module library by utility routine LMR. This is necessary because the COBOL compiler does not explicitly test this property at the present time.

For identification purposes, the ANSICOB compiler creates a linkage editor control card (OVERLAY...) at the beginning of a module. During the LMR run these cards are removed with a corresponding notification. This is necessary, and therefore appropriate, for further processing by the Dynamic Linking Loader (DLL). Only the Dynamic Linking Loader (DLL) is capable of loading shared code and must in this case be used. It is called with

```
/EXEC   (modulename, libraryname) or
/LOAD   (modulename, libraryname)
```

Note, however, that shared modules are only loaded into the system's class 4 memory, where they are available to all tasks, if the system administrator has entered them into the appropriate system module tables during initiation of the BS2000 session (cf. "System Controller's Guide" SHARE command).

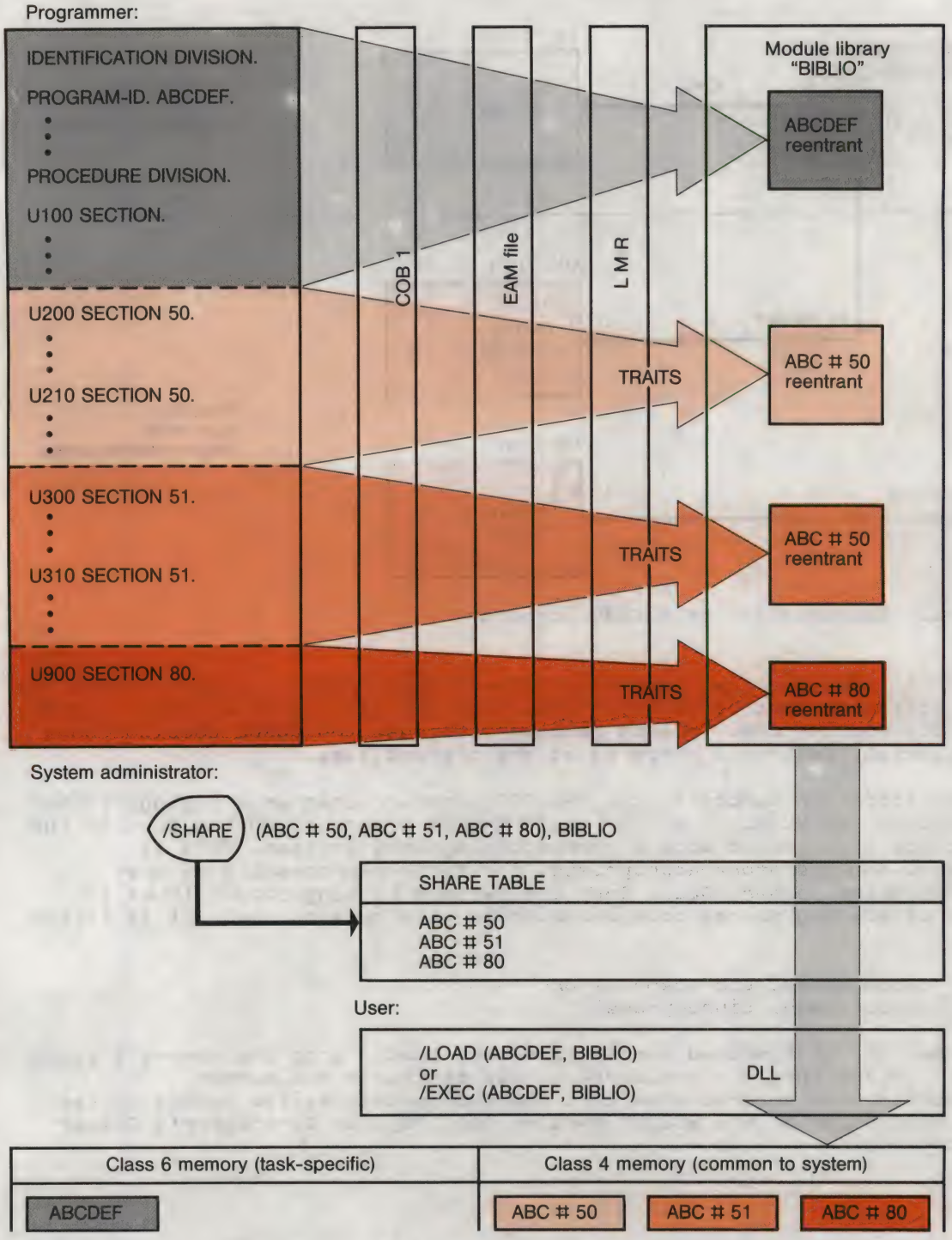


Fig. 4-11 Generation of shared code

4.2.5 Use of Other System Interfaces

There are no limitations on a DCAM COBOL program within the framework of COBOL use in BS2000.

Attention is drawn to the possibility of using the SHARED UPDATE mode for the USER, PAM and ISAM access methods of the data management system. This mode provides the necessary protection mechanisms for access of a task group to a file (see manual "DVS").

4.3 EXECUTION OF A DCAM PROGRAM: DCAM TASK

At first a DCAM task is just any BS2000 task. As soon as a program opens a DCAM application within this task, the task attribute is set to "TP" (transaction processing).

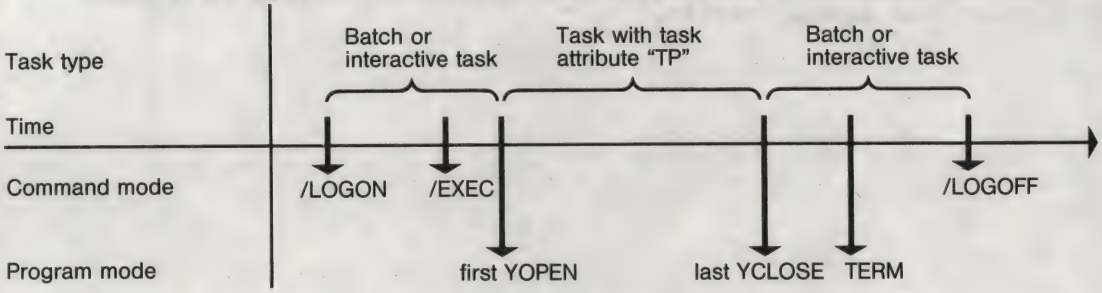


Fig. 4-12 Task types

4.3.1 Starting a DCAM Task

Before a task can become a DCAM task, it is started as a batch or interactive task with the /LOGON command (see "Control System Command Language"). This command can be issued

- for **batch tasks on punched cards** which are input locally (by means of punched card readers) or remotely (by means of 8418 Remote Terminals).
- for **batch tasks** from other tasks by means of **ENTER files** which contain a /LOGON command at the beginning. Batch tasks are started by the system if the resources are available and the limitation defined by the system administator allows it.
- for **interactive tasks** from any **interactive terminal**, e.g. an 8161 Data Display Terminal. In this case, the task is started immediately.

Note:

The virtual connection between a terminal and an application is set up as early as during the predialog (see "Terminal Access Support"). In the case of interactive tasks, this is the connection between the terminal and the "$DIALOG" application. After such a task has become a DCAM task, the connection to the terminal remains in existence and it is thus not possible to set up a connection from this terminal to the DCAM program controlling the task. However, it is possible to test and modify the program from this terminal using IDA (see Fig. 4-13).

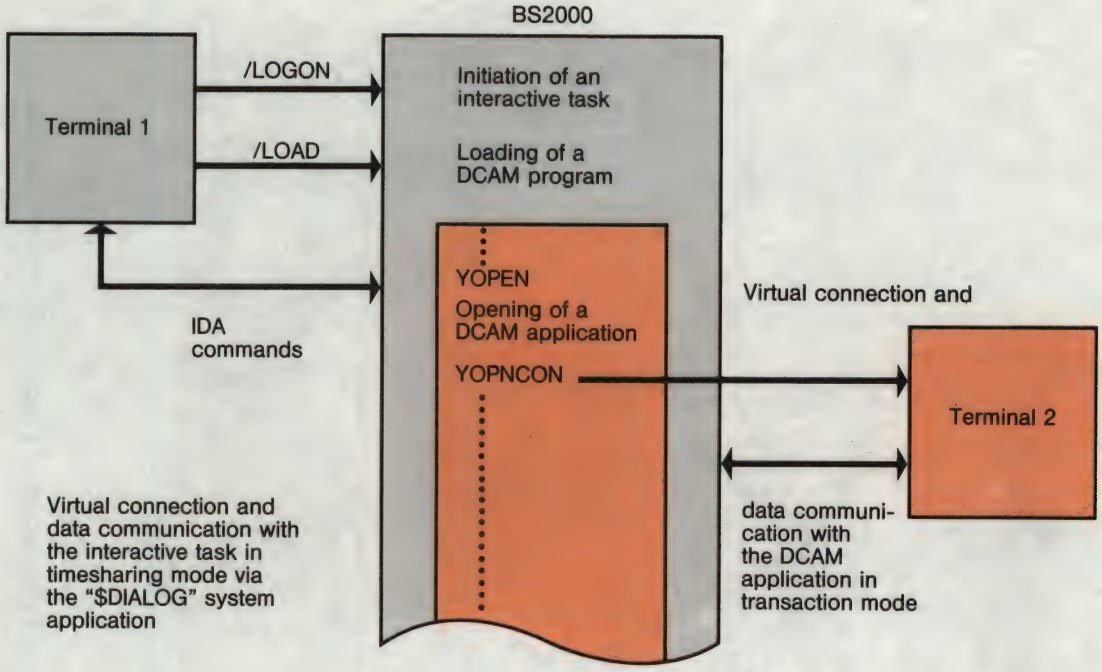


Fig. 4-13 Timesharing and inquiry and transaction processing in a task

4.3.2 Termination of a DCAM Task

A DCAM task is returned its original task attribute when the program controlling it closes the last DCAM application or is terminated. It is then managed as a batch or interactive task as before and is terminated with the /LOGOFF command. This command can be issued for

- a batch task on a punched card in a card deck
- as the last command in an ENTER file
- as input from an interactive terminal
- in the program mode by means of the LGOFF macro or indirectly by means of the TERMJ macro if a branch is made to /LOGOFF.

Tasks can also be terminated by means of

- the operator's /CANCEL command or
- the /CANCEL command in the interactive task for tasks with the same user ID, but different TSNs,
- the operator's /SHUTDOWN command, or
- by abnormal task termination in the case of a system error.

A APPENDIX

A.1 DCAM CALLS

Types of DCAM call:

- Macro calls (ASSEMBLER = A)
- CALL statements (COBOL = C)

Type	Call	Function	Description
A	YACB	*)	Generates an application control block.
A	YAPPL	Name assignment	Stores parameters of DCAM application in CLT, and also deletes same.
A	YCCB	*)	Generates a connection control block.
A; C	YCHANGE	Connection	Change characteristics of a connection already established.
A; C	YCLOSE	Existence	Closes down a DCAM application.
A; C	YCLSCON	Connection	Cancels a request or closes down a virtual connection.
A	YCONN	Name assignment	Stores connection parameters in CLT, and also deletes same.
A	YDCG	*)	Generates a distribution code group block.
A	YDDACB	*)	Generates a (dummy) section for control block ACB
A	YDDCCB	*)	Generates a (dummy) section for control block CCB
A	YDDDCG	*)	Generates a (dummy) section for control block DCG
A	YDDDIP	*)	Generates a (dummy) section for control block DIP
A	YDDENB	*)	Generates a (dummy) section for control block ENP
A	YDDRPB	*)	Generates a (dummy) section for control block RPB
A	YDIP	*)	Generates a distribution parameter block.

DCAM Calls

Type	Call	Function	Description
A	YENB	*)	Generates an event notification block for relating asynchronous messages to contingency routines.
A; C	YFORBID	Data communication	Cancels the assignment of a distribution name to a distribution code group.
A	YMODCB	*)	Changes items in existing control blocks.
A; C	YOPEN	Existence	Opens a DCAM application.
A; C	YOPNCON	Connection	Sets up a virtual connection.
A; C	YPERMIT	Data communication	Assigns a distribution name to a distribution code group.
A; C	YRECEIVE	Data communication	Receives a message, express message or transport acknowledgment.
A; C	YREJLOG	Connection	Rejects a connection request.
A C	YRESET	Data communication	Deletes receive calls; Changes the connection state CS/CA.
A	YRPB	*)	Generates a request parameter block.
A; C	YSEND	Data communication	Send a message or express message.
A	YSENDREC	Data communication	Combines sending a message or express message with receiving a message, express message or transport acknowledgment
A	YSET	*)	Sets a switch, which results in control block macros being executed in PI status
A; C	YSETLOG	Existence	Changes the status of an application.
A	YSHOWCB	*)	Transfers individual items from a control block into the user area.
A; C	YTESTCB	*)	Compares an item in a control block with a given value.
C	YWAIT	*)	Waits for termination of asynchronous CALLs.
A	YGENCB	*)	Generates one or more control blocks of any kind.
A; C	YINQUIRE	Existence; connection	Retrieves information on applications and virtual connections.

*) Control block functions

A.2 LIMIT VALUES

- **Asynchronous processing**

The following table shows how many asynchronous instructions may be processed concurrently.

Maximum number	Valid for call of type	Function
8 per application	YOPNCON ACCEPT ANY	Accept request from any partner
8 per application	YOPNCON ACCEPT SPEC	Accept request from a specific partner.
8 per application	YOPNCON ACQUIRE	Issue request
8 per task of an application	YRECEIVE ANY	Receive a message or transport acknowledgment from any partner
8 per virtual connection	YRECEIVE SPEC	Receive a message or transport acknowledgment from a specific partner.

- **Use of distribution codes**

The following table shows the maximum values for the definition and assignment of distribution codes.

Type	Upper limit	Applicable to
Number of distribution code groups per virtual connection	8 16	COBOL ASSEMBLER
Distribution codes per code group	8	ASSEMBLER, COBOL
Tasks per distribution name	8	ASSEMBLER, COBOL

- The number of concurrently active DCAM applications which have been opened by the same task is limited. The precise quantity is to be found in the relevant release notice for the BS2000 operating system.

  The number of "non-predefined applications" is also limited (see manual "Network Management in BS2000", DCSTART command). This value can relate to a task or the system as a whole. These limiting values are defined at the system start and can be modified during operations.

  An application can maintain only a certain number of connections simultaneously. There is also a limit to the number of connections a "non-predefined application" can employ (see above).

Limit Values

- **Maximum message Length (MAXLN)**

  Local parameter which influences the economy of the buffers provided by the system. It contains the maximum length of the data being sent (Transport Service Data Unit, TSDU).

  When EDIT = USER:
  1 message = 1 TSDU (YSEND)

  When EDIT = SYSTEM, EDITOUT = PHYS or FORM:

  DCAM transmits the system-edited message in segments, the size of which is determined by MAXLN and the device capacity. It is up to the user to ensure the device capacity is not exceeded.

  Any records longer than MAXLN are truncated during editing.

  Maximum length for a user message per YSEND: 32767 bytes

  Note:

  An edited record is always longer than user data as control characters are converted and protocol labels added.

  The following table shows the maximum values of MAXLN:

<table>
<tr><td>Requested MAXLN.</td><td colspan="2"><— 65530</td><td colspan="2">Default value</td></tr>
<tr><td>DCAMVER</td><td>8.0</td><td>8.0</td><td colspan="2">Default value</td></tr>
<tr><td rowspan="2">MAXLN</td><td rowspan="2">x = 65530(*)</td><td>4096</td><td colspan="2">4096</td></tr>
<tr><td>with FEP
with DXC</td><td>4096</td><td>32767</td></tr>
</table>

  (*) The results depend on the HW/SW configuration and generation.

- **Limit values for resources**

  The following static maximum values apply in DCAM V8.0:

Applications in DCAM	255
GID/application (permitted)	32
GID/application (not permitted)	32
DID/application	32

The number of applications + the number of P1 events is restricted by the BS2000 Name Manager to around 250 per task.

In BS2000 the number P1 contingencies generated for a task but not yet executed is limited (the maximum number in BS2000 V7.5 is 400). It may be possible to modify this value by means of a REP in BS2000 (cf. DCAM release notices).

A.3 SETTING UP A CONNECTION FROM A TERMINAL

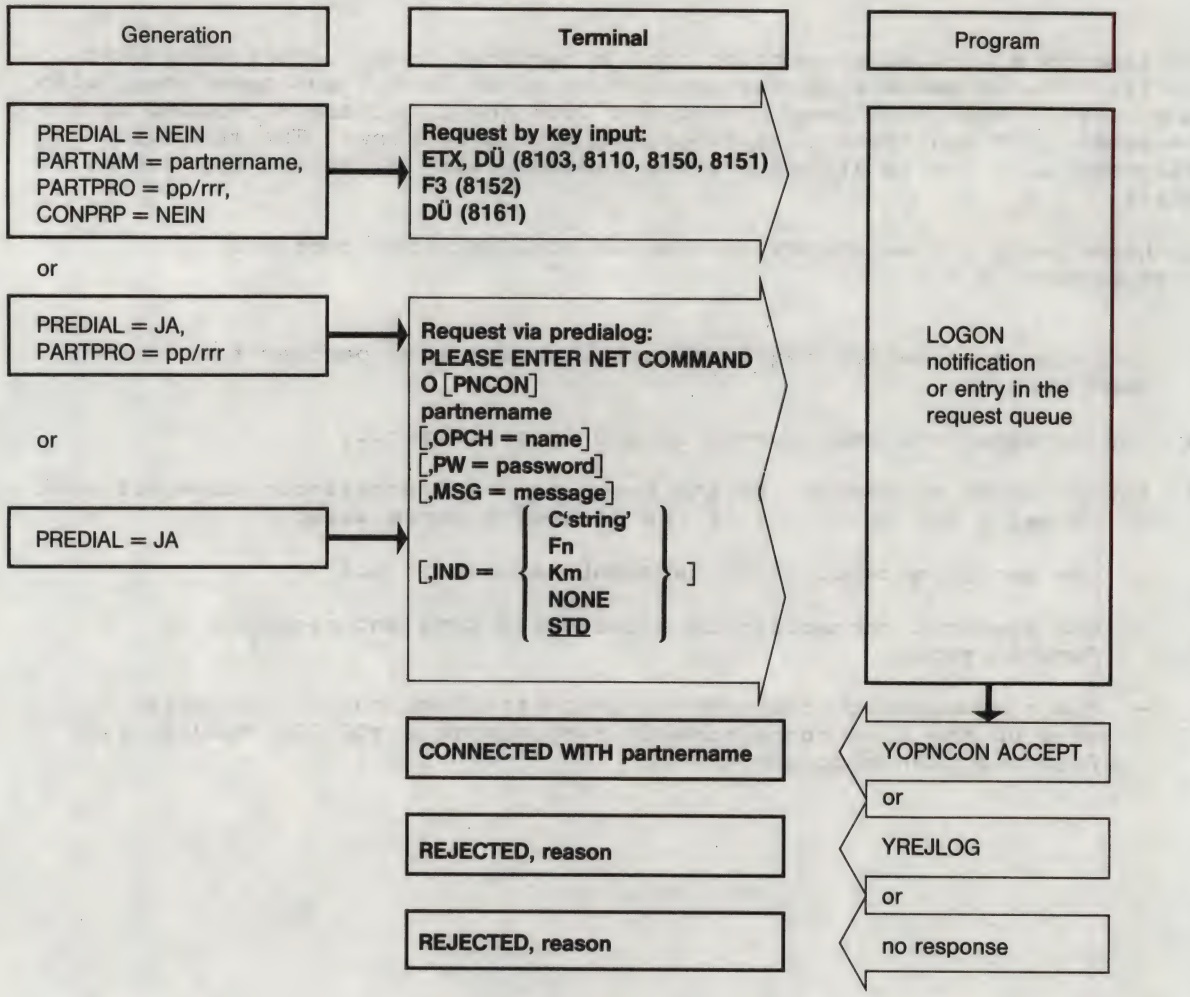
Setting up a virtual connection from a terminal is an undertaking which varies in accordance with the way the terminal itself was generated, with the options that have been preset and with the constraints imposed by the relevant DCAM application as the communication partner. The following diagrams will try to illustrate two fundamentally different approaches to this:

In both cases the existence of the two communication partners is presupposed, i.e.:

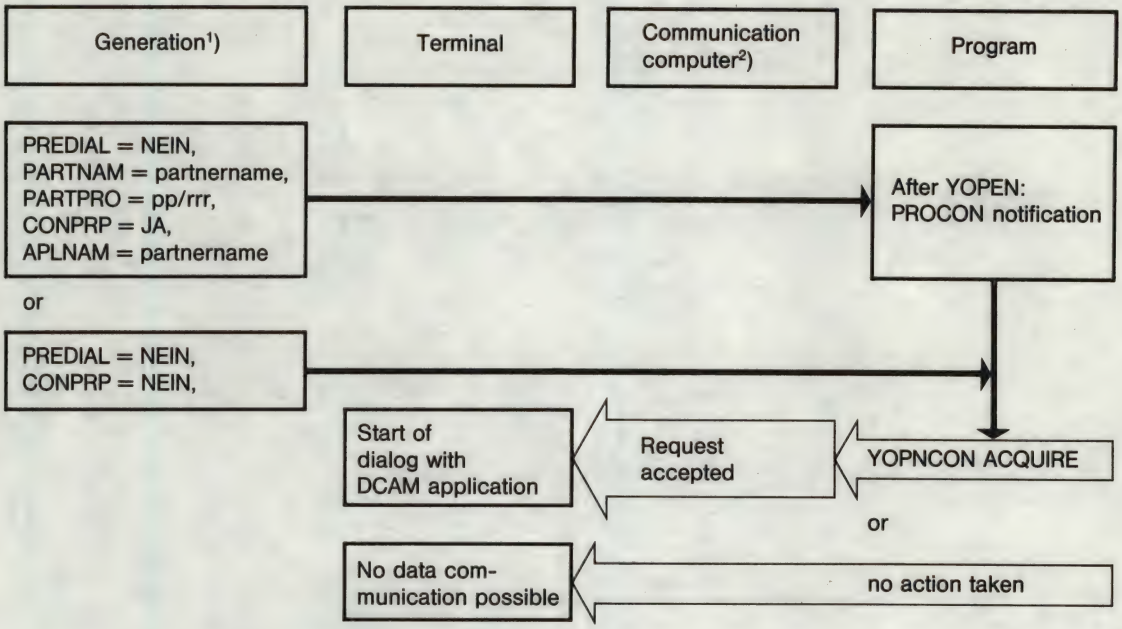
- the DCAM application required as the prospective partner has already been opened;
- the terminal has been turned on and is operational;
- the physical connection at the level of the transmission line has been or is being set up in one of the following three ways:
  - the person attending the terminal has set it up;
  - the physical connection is a dedicated line and requires no further setup;
  - the communication computer to which the terminal is connected sets up the line connection in response to a YOPNCON ACQUIRE call from the DCAM program.

Setting up a connection from a terminal

Initiative lies with the terminal (DCAM application with LOGON attribute):



Initiative lies with the DCAM application (opened with NLOGON attribute):



[1]) see XSTAT macro in "Generation of a Data Communication System"
[2]) Communication computer must have been activated with /BCIN and /BCACT commands (see "Network Management in BS2000")

GLOSSARY OF TRANSDATA TERMS

Cross-references are underlined

A

Administration: [TRANSDATA] Administration

Administrator: [TRANSDATA] Administrator

Administration center Partner to the administrator. Controls operations by passing on jobs to the administration managers.

Administration console Facility connecting the administrator with the administration center.

Administration manager Software module in the host and communication computer which executes jobs from the administration center and reports back to it.

C

[Communication] access system That set of software components of an operating system which offers the applications interfaces to the communication system.

[Communication] application Facilities for processing data exchanged between communication partners.

[Communication] application program Processing specification for controlling the communication application. It uses the interfaces of the communication access system.

Communication computer Computer used to set up teleprocessing systems.

Communication partner [TRANSDATA-] communication partner

[Communication] protocol Description of the conditions and formats relating to data transfer between similar functional levels in the data communication system (user services, transport services, link services).

Computer (ISO 2382)

Connection request Request from a communication partner to the data communication system to set up a virtual connection with another communication partner.

D

Data

A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by humans or automatic means. ISO

Data communication

Data transfer between data source and data sink via one or more data links according to a link protocol. ISO-E

Data communication system

Complex facility comprising hardware and software products which enables two or more communication partners to transmit data in accordance with specific rules.

Data flow control

Capacity control along the path of the message through the data transmission network.

Data processing system

(ISO 2382)

Data sink

That part of a data terminal equipment that receives data from a data link. ISO-E

Data source

That part of a data terminal equipment that enters data into a data link. ISO-E

Data transmission network

Sum total of all the hardware and software facilities enabling physical transfer of data from the data source to the data sink.

[DCAM] application

Communication application which is controlled by at least one DCAM application program and generated during one or more DCAM tasks.

[DCAM] application program

User program which uses the DCAM access method and controls the DCAM application.

[DCAM] connection function

DCAM functions expressed by instructions and notifications which are concerned with establishing and clearing down virtual connections.

[DCAM] control block function

DCAM interface instructions for the generation and manipulation of control blocks. All DCAM macro calls relate to these control blocks.

[DCAM] data transmission function

DCAM functions expressed by instructions and notifications which are concerned with transmitting and receiving messages and acknowledgments.

[DCAM] distribution code

A defined character string within an input message enabling distribution of the message within a DCAM application; the location in the message is optional but its length is restricted.

[DCAM] event

One of a number of DCAM events used for coordinating certain operations in the data communication system. It is time-independent of program execution (asynchronous event)

[DCAM] existence function

DCAM functions expressed by instructions and notifications which are concerned with creating and canceling DCAM applications.

[DCAM] name assignment function

DCAM functions expressed by instructions or commands which allow the user to write the user programs without regard to current parameter values such as DCAM application name, partner name, etc.

[DCAM] status function

DCAM functions expressed by an instruction which enable a DCAM application to obtain information about itself or communication partners.

[DCAM] task

A task which has opened a DCAM application through an explicit DCAM call (primary task) or has identified itself to an existing DCAM application (secondary task).

Dialog message

Message which requires a response, or is itself a response.

In the case of UTM the first message is the inquiry by a TRANSDATA communication partner. This serves either to start or continue a transaction.

Dialog step

Part of a transaction. It begins with a dialog message to a TRANSDATA communication partner, and ends with the associated response.

Digital data processing system

A data processing system which, seen as a functional unit, is a sequential circuit.

E

Express message

Message of restricted length to a DCAM application or a terminal transmitted with a higher priority than normal messages.

F

Format terminal

Type of virtual terminal. The data structure is formed by fields with differing characteristics.

Front-end processor (FEP)

Local communication computer which is connected directly to the I/O channel of the host computer.

H

Host computer

Digital data processing system whose primary purpose is the processing of data in application programs.

I

Instruction

An imperative statement which, in the context of a given programming language, cannot be divided into other imperative statements.

L

Line terminal

Type of virtual terminal. The data structure is formed by lines.

Link services

Services comprising node services and the port services which relate to other processor nodes.

M

Message

(ISO-2382)

N

Node services

Services for handling messages to and from processor nodes.

P

[PDN] application — Communication application in the communication computer which uses the interfaces of the TRANSDATA PDN communication access system.

Port — Facility concerned with the data transfer between a processor node and its environment.

Port services — Services provided for the transfer of data between a processor node and its environment (communication applications), local peripherals, terminals or other processor nodes).

Primary task — Task which opens a DCAM application and defines its characteristics.

Processing subsystem — Subgroup of components in the teleprocessing system, consisting of a terminal computer and terminals.
It constitutes a self-contained subsystem within a distributed processing network.

Processor node — Functional facility in the host computer or communication computer which provides the transport services. It can be addressed from throughout the network.

Program — (ISO-2382)

R

Region — Part of the data communication system consisting of one or more processor nodes which, for the purpose of addressing, are regarded as one by the transport services.

Remote front-end processor (FEPR) — Remote communication computer which is not directly connected to a host computer and whose function is limited to data communication.

S

Secondary task

Task linking itself to an opened DCAM application and sharing its resources.

Shareable DCAM application

DCAM application which can be used simultaneously by several DCAM tasks.

Station

A terminating element of the data communication system as viewed by the transport services which is addressable from any point in the network.

Station services

Services which simplify data communication by handling messages transferred to and from communication partners.

T

Teleprocessing system

A system in its entirety, including data processing and data communication functions.

Terminal

The set of functional units that consists of the data terminal equipment and the data circuit-terminating equipment, and their common interface. ISO-E

Terminal user

Person who uses a terminal in order to exchange data with a TRANSDATA communication partner.

Terminal computer

Communication computer to which terminals are connected. On it are run communication application programs for terminal control and data processing.

Transaction

Processing of a self-contained job issued to the transaction application by the user. The user can be:
- a terminal user
- a different UTM application
- a TRANSDATA communication partner.

The transaction may consist of one or more dialog steps. The requisite resources are assigned to the transaction.

[Transaction] application

Communication partner to which transaction jobs are directed and which processes the desired transactions.

[Transaction] application program

Processing specification for the control of the application. It uses the program interface of the transaction system.

[Transaction] job

Instruction directing the application to execute a transaction.

[Transaction] code /TAC

Information for the control of a transaction. In the case of UTM the TAC serves to select the application program units.

[Transaction] session

Period of time during which a communication partner can issue jobs. This period is delimited by the granting and withdrawal of certain privileges.

Transaction system

Those software components which constitute the interface between application and environment. It controls and supervises transactions. It avails itself of the services of the operating system, in particular the communication access system and data base system.

[TRANSDATA] administration

Startup, control and management of the TRANSDATA components of a teleprocessing system.

[TRANSDATA] administrator

Person or program responsible for the administration.

[TRANSDATA] communication partner

Facilities which maintain virtual connections and exchange data.

Transport acknowledgment

Notification indicating completion or abnormal termination of data communication.

Transport services

Services for the exchange of data between communication partners. Initiates and monitors the movement of messages through the data transmission network and manages virtual connections.

U

Unsolicited message

A message which neither requires a response and nor is itself a response. If a terminal user sends such a message to an application program unit, he will receive a standard acknowledgment from UTM. If another TRANSDATA communication partner sends such a message, no standard acknowledgment is received from UTM.

User services

Services which serve to transmit messages for one of the two communication partners. The user services comprise the station services and the port services which relate to the communication partner concerned.

[UTM] application

Communication partner to which terminal users, UTM applications and other TRANSDATA communication partners can direct transaction jobs. The application processes the desired transactions. After being generated it is controlled by the application program.

[UTM] application program

Processing specification for the control of the UTM application. It uses the UTM program interface and consists of the UTM linkage program and the user's program units.

[UTM] [application] program unit

Self-contained part of a UTM application program which processes a sub-problem of an application. It is addressed by means of the transaction code. UTM activates the program unit when there is a message for it. A program unit carries out one dialog step at the most.

[UTM] dialog step

Part of a transaction. It commences with a dialog message sent by the terminal user or another TRANSDATA communication partner to an application and terminates with the response from that application.

[UTM] linkage program

UTM application program unit prepared by the system. All that the user does is define its scope. This program serves to connect the application program to the UTM. Every application program includes such a linkage program.

[UTM] terminal

Conceptual terminal which exists in name only. It is defined at the generation of the UTM application and decouples the latter from the data communication network. A UTM terminal can stand for several physical terminals and a number of UTM terminals can stand for just one physical terminal. A UTM terminal can also stand for any TRANSDATA communication partner. The allocation may be altered during operation by means of an administration command.

[UTM] transaction

Processing of a self-contained job by the application, which for this purpose can use one or more program units.
A transaction can consist of one or more dialog steps. Resources, such as storage space etc., are allocated to it.

[UTM] [transaction] job

Instruction directing a UTM application to execute a transaction. It includes a transaction code and, if applicable, the data to be processed. It is submitted by the terminal user, by UTM program units of the same or a different application, or by other TRANSDATA communication partners.

[UTM] [transaction] session

Period of time during which a terminal user, a different UTM application or a TRANSDATA communication partner can submit certain jobs to a UTM application. It commences when the relevant authorization is given and ends when this authorization is withdrawn.

V

Virtual connection

Pairing of two communication partners which enables them to exchange data.

Virtual terminal

Conceptual representation of a terminal whose functions are mapped onto the physical characteristics of different terminals.

LITERATURE

COB1
COBOL Compiler
Reference Manual

COB1
COBOL Compiler
User's Guide

Software product
COLUMBUS-ASS
User's Guide

BS2000
Control System
Command Language
Reference Manual

TRANSDATA
BS2000
DCAM
COBOL Calls
User's Guide

TRANSDATA
BS2000
DCAM Macros
User's Guide

TRANSDATA
BS1000, BS2000, PDN
Directions for the Terminal User
User's Guide

BS2000
DMS
Disk Processing
Reference Manual

BS2000
Executive Macros
Reference Manual

TRANSDATA
BS2000
FHS
Format Handling System
User's Guide

TRANSDATA
BS1000, BS2000, PDN
Generation of a
Data Communication System
User's Guide

TRANSDATA
BS2000, PDN
Network Management in BS2000
User's Guide

BS1000, BS2000, DPS, PDN
TeLeprocessing
TRANSDATA Software
Reference Guide

BS2000
UtiLity Routines
Reference Manual

U1786-J-Z75-1-7600

TRANSDATA
DCAM
Program Interfaces
Reference Manual

Edition January 1985
(Software Product DCAM V8.0A) (BS2000)

Order No. U1786-J-Z75-1-7600

This manual is part of the overall system description of the 7·500/7·700/4004 and TRANSDATA systems. Your comments, as well as any suggestions which you might have, will help us greatly in improving the quality of this manual. Special attention should be paid to those requirements which have practical significance when answering the following questions:

- Where is there too much or not enough detail?
- Where should we provide more supporting diagrams?
- Where is the text difficult to understand?
- Where could the basic structure of the manual be improved?

Please use the back of this page for your suggestions. Give specific page references where appropriate or enclose pages copied from the manual.

Please fill the following if you require a reply.

From:

Name _____

Company/Department _____

Address _____

Telephone _____ Date _____

Your Siemens office _____ Mr/Mrs/Miss _____

Siemens AG
D ST PM 2
Manualredaktion

Otto-Hahn-Ring 6

8000 München 83

Re: Ideas, suggestions, corrections

I am using this manual for

( ) General introduction
( ) Teacher's guide
( ) Student guide
( ) Reference guide
( ) _____
_____ (please state)

I am concerned with:

( ) System operation/administration
( ) System administration
( ) Programming
( ) Systems analysis
( ) Applications engineering
( ) Teaching
( ) _____
_____(please state)

Page	Remarks

TRANSDATA
DCAM
Program Interfaces
Reference Manual

Edition January 1985
(Software Product DCAM V8.0A) (BS2000)

Order No. U1786-J-Z75-1-7600

This manual is part of the overall system description of the 7·500/7·700/4004 and TRANSDATA systems. Your comments, as well as any suggestions which you might have, will help us greatly in improving the quality of this manual. Special attention should be paid to those requirements which have practical significance when answering the following questions:

- Where is there too much or not enough detail?
- Where should we provide more supporting diagrams?
- Where is the text difficult to understand?
- Where could the basic structure of the manual be improved?

Please use the back of this page for your suggestions. Give specific page references where appropriate or enclose pages copied from the manual.

Please fill the following if you require a reply.

From:

Name _____

Company/Department _____

Address _____

Telephone _____ Date _____

Your Siemens office _____ Mr/Mrs/Miss _____

Siemens AG
D ST PM 2
Manualredaktion

Otto-Hahn-Ring 6

8000 München 83

Re: Ideas, suggestions, corrections

I am using this manual for

( ) General introduction
( ) Teacher's guide
( ) Student guide
( ) Reference guide
( ) _____
_____ (please state)

I am concerned with:

( ) System operation/administration
( ) System administration
( ) Programming
( ) Systems analysis
( ) Applications engineering
( ) Teaching
( ) _____
_____(please state)

Page	Remarks

Published by Bereich Datentechnik
Postfach 830951, D-8000 München 83

Siemens Aktiengesellschaft